

4

AD-A197 039

DTIC FILE COPY

ADAPTIVE MACHINE VISION  
ANNUAL REPORT

**SAIC** Science Applications International Corporation

DTIC  
ELECTE  
JUL 07 1988  
S H D

DISTRIBUTION STATEMENT A

Approved for public release;  
Distribution Unlimited

88 7 06 134

ADAPTIVE MACHINE VISION  
ANNUAL REPORT

By  
William W. Stoner  
Michael H. Brill  
Doreen W. Bergeron

DTIC  
SELECTE  
JUL 07 1988  
S H D

8 March 1988

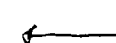
SCIENCE APPLICATIONS INTERNATIONAL CORP.  
6 Fortune Drive  
Billerica, MA 01821  
(617) 667-6365

DISTRIBUTION STATEMENT A

Approved for public release  
Distribution Unlimited

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle)  Adaptive Machine Vision		5. TYPE OF REPORT & PERIOD COVERED  Annual Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William W. Stoner Michael H. Brill Doreen W. Bergeron		8. CONTRACT OR GRANT NUMBER(s)  N00014-86-C-0601
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS INTERNATIONAL CORPORATION 6 Fortune Drive Billerica, MA 01821		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Strategic Defense Initiative Organization Innovative Science and Technology Office, Washington, D. C.		12. REPORT DATE 8 March 1988
		13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Boston Detachment 495 Summer St., Boston, MA 02210-2109		15. SECURITY CLASS. (of this report)  Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Machine Vision, Pattern Recognition, Retinal Model		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) → The SDI missions of coarse track, stereo track and discrimination are examined from the point of view of a machine vision system. (KR) 		

Concepts for stereo vision are examined and both optical and digital implementations are discussed. The requirements of a pattern recognition system with tolerance to shift, scale and aspect angle are examined. The Neocognitron architecture of Fukushima is discussed, and a scaling law is derived for the Neocognitron.

A retinal model with adaptive contrast and adaptive spatial resolution is presented. This retinal model, called IRIS, has properties which would be helpful in the front-end of a machine vision system. The proposed retinal model adjusts the set point of dynamic range to provide maximum contrast about the average scene intensity. It also adapts its spatiotemporal resolution to the local scene intensity, providing more averaging at low intensity, thereby combating photon noise.

# Table of Contents

SECTION	TITLE	PAGE
1.0	INTRODUCTION.....	1
1.1	Adaptive Machine Vision for Boost Phase.....	1
1.2	Adaptive Machine Vision for Post Boost Phase....	4
1.3	Adaptive Machine Vision for Midcourse Defense...	7
1.4	Adaptive Machine Vision for Terminal Defense....	10
2.0	Applications of Machine Vision to Stereo Tracking....	12
2.1	Optical Implementation.....	18
2.2	Digital Implementation of Stereo Algorithm.....	24
2.2.1	Computational Load of the Zero-Crossing Stereo Algorithm.....	25
2.2.2	Computational Load of the Cepstral Stereo Algorithm.....	31
2.3	Stereo Algorithm Comparison.....	35
3.0	Neocognitron Evaluation.....	37
3.10	Pattern Recognition with Tolerance to Shift, Scale and Aspect Angle.....	40
3.11	Alternative Approaches to Invariant Pattern Recognition.....	43
3.2	Computational Load of the Neocognitron.....	45
3.2.1	Scaling of the Neocognitron.....	46
3.2.2	Operation Count for One Neocognitron Cycle.....	49
3.2.2.1	Connections Between Modules.....	49
3.2.2.2	Connections Within Modules.....	52
3.2.2.3	Nonlinear Combining Within Modules.....	53
3.2.2.4	Nonlinear Combining Between Modules.....	56
3.2.2.5	Conclusions on Neocognitron Computational Load.....	59
4.0	AC-coupled Retina with Cooperative Receptors.....	61
5.0	Bibliography.....	72
6.0	Appendix - Parallel Implementation of IRIS.....	A-1



Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## 1.0 INTRODUCTION

*the Strategy Defense Initiative*

The mission of SDI is to develop defenses against threatening ballistic missiles. There are four distinct phases to the SDI defense; boost, post boost, midcourse and terminal, ~~as shown in Figure 1.~~ In each of these phases, one or more machine vision functions are required, such as pattern recognition, stereo image fusion, clutter rejection and discrimination. *In 2. - correct*

### 1.1 Adaptive Machine Vision for Boost Phase

Consider the Acquisition, Tracking, Pointing and Fire Control (ATP-FC) function during boost phase. The hot rocket plume provides a bright signature in the mid and long wave infrared. However, the centroid of the thermal plume image lies anywhere from 50 to 300 meters behind the booster, and this distance is as much as 10x the length of the missile body. The stressing problem is not plume detection, but accurate tracking of the actual missile, given that the size and shape of the plume evolves constantly during ascent, and changes abruptly between stages. Figure 2 provides a guide to these variations in the plume dimensions. The plume size and shape is changing dynamically, and these changes result in large variations in the offset from the thermal image centroid of the plume to the hard body. Even if the changes in plume shape with aspect angle could be neglected, these dynamic and sometimes abrupt changes in signature pose a difficult, unsolved problem of ATP-FC, one

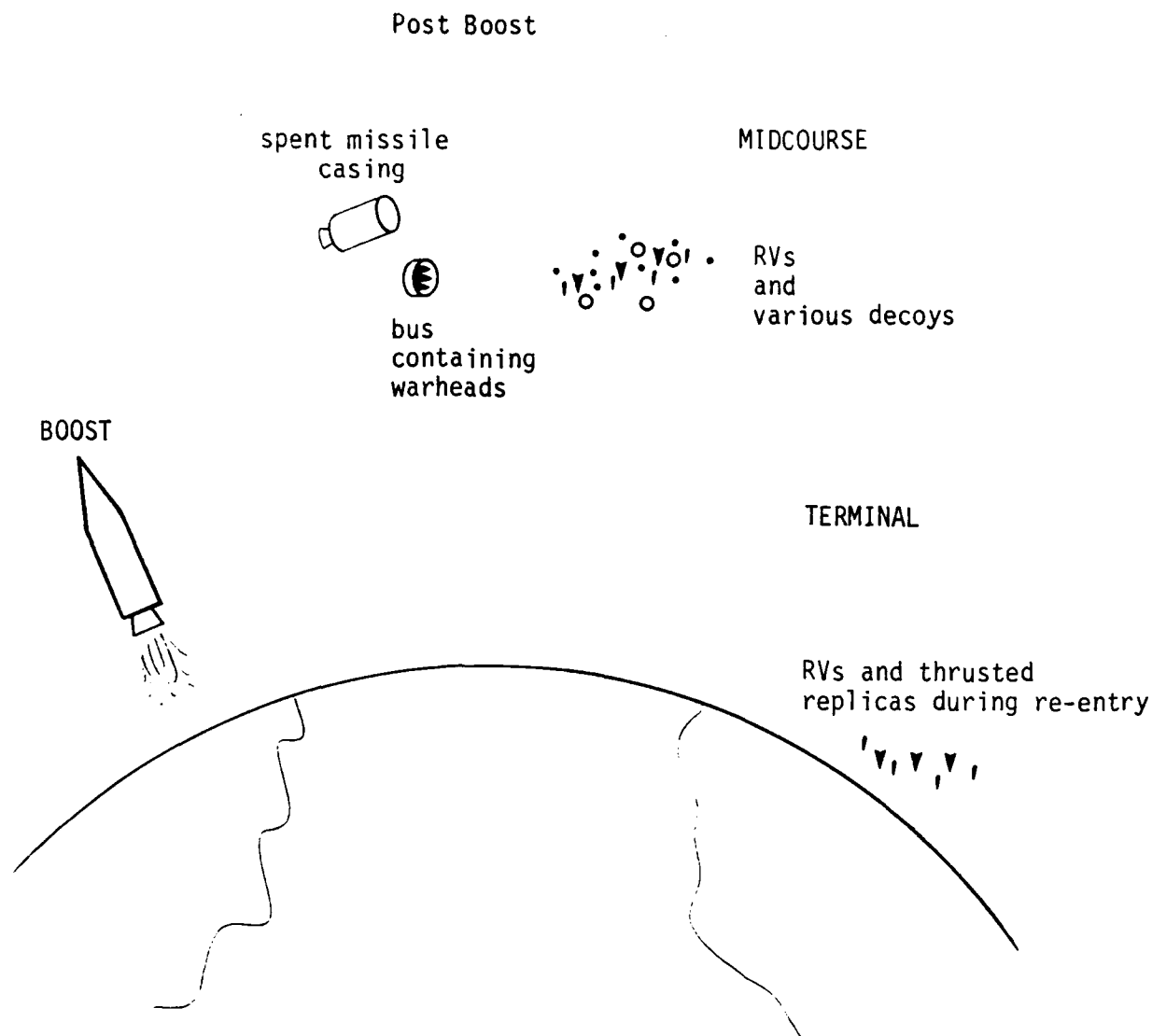


Figure 1. The SDI layered defense concept provides four defensive zones, boost, post boost, midcourse and terminal. During the boost phase, the SDI functions requiring adaptive machine vision support acquisition, track, aimpoint selection and kill assessment; during post boost, discrimination and track; During the midcourse and terminal phases, acquisition, tracking, discrimination and kill assessment.

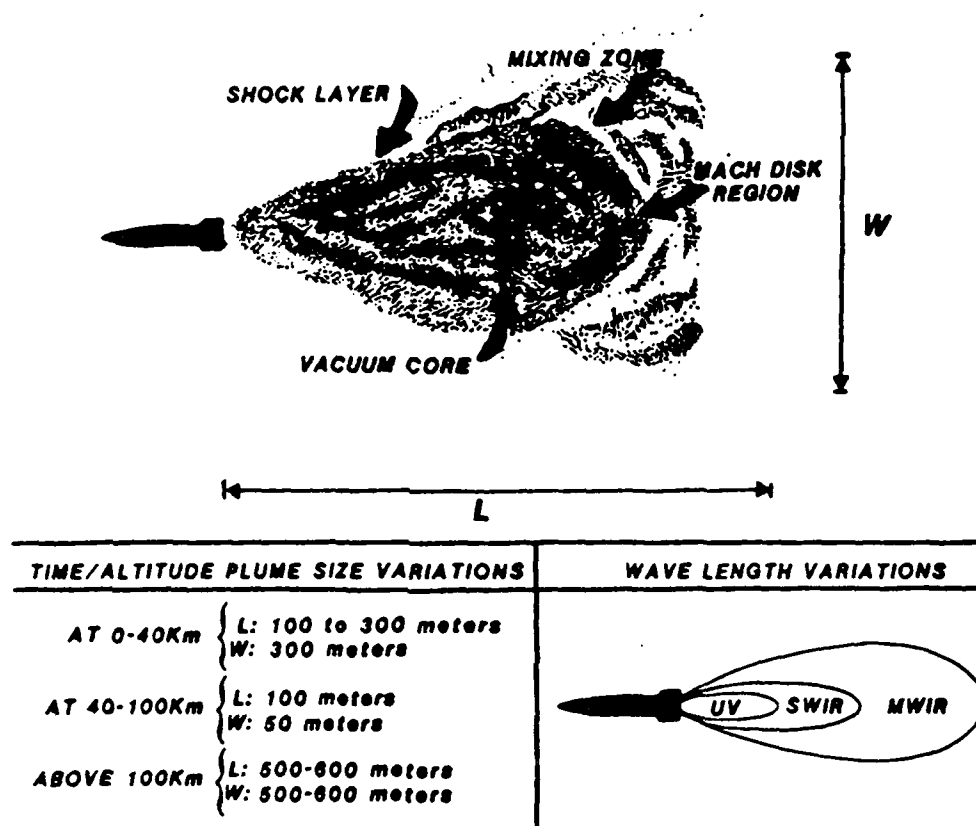


Figure 2. The hot missile plume provides a good strong signature for detection. However, the thermal plume centroid is located from 50 to 300 meters from the missile body. Plume to hardbody handover is a critical problem which cannot be solved by simple centroid tracking or rigid template correlation tracking. An adaptive approach is required.



that requires an adaptive pattern recognition capability.

The fusion of imagery from separate space surveillance platforms, as depicted in Figure 3 is another challenging area of adaptive machine vision for SDI. It is known from studies of mono vs. stereo tracking that stereo tracking with two passive angles-only sensors can provide much smaller error ellipsoids for the target position than is possible with a single sensor of comparable size performing monostatic angles-only tracking.

The fine-track function is performed quite well by a Kalman filter, once coarse stereo tracking has been achieved. This involves associating the target images from the separate stereo sensors, as well as estimating the offsets from the plume to the hard body. The key steps of this preliminary coarse stereo track function is frame-to-frame association at the individual sensor level, and stereo fusion of the images from the separate sensors. These functions are performed naturally by biological visual systems, inspiring our efforts to draw upon theories of stereopsis in humans to develop a comparable capability in machine vision.

## 1.2 Adaptive Machine Vision for Post Boost Phase

In the post boost phase, the bus containing the threat weapons (re-entry vehicles or RVs) makes a series of thrusting accelerations to independently launch each RV towards a

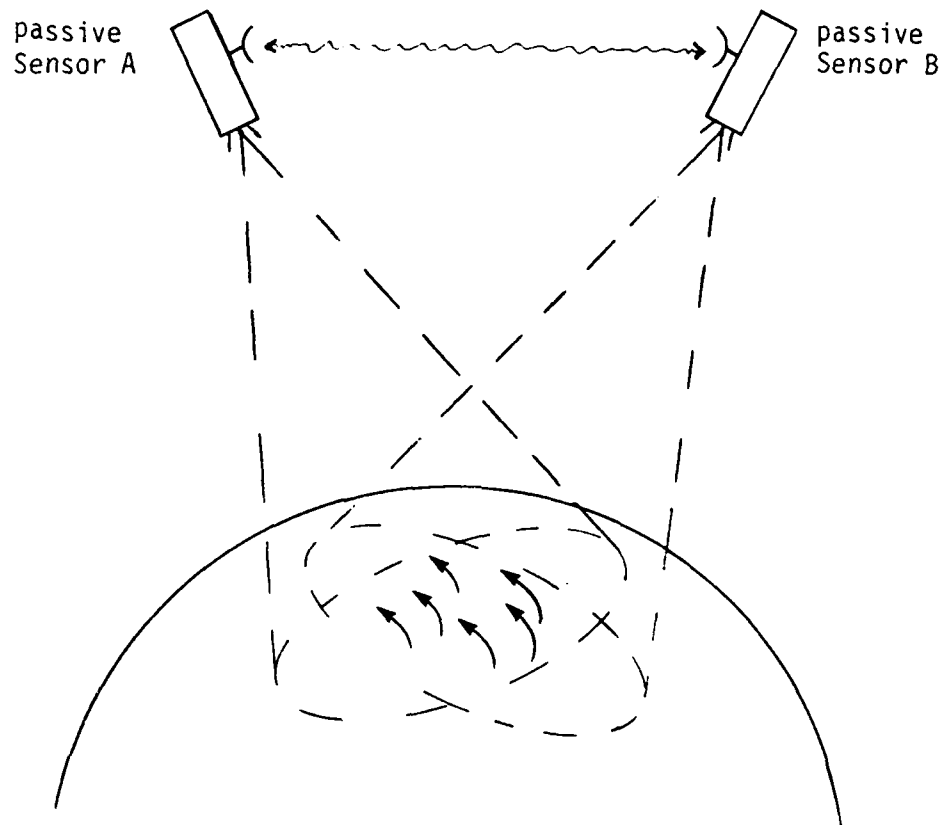


Figure 3. Trajectory estimation is greatly enhanced by stereo tracking. The first step in stereo tracking is the proper association of targets seen by sensor A with the targets seen by sensor B. This target association problem grows in complexity as  $N^2$  for stereo tracking, where  $N$  is the number of targets. For  $m$  sensors and  $N$  targets, the computational complexity grows as  $N^m$ . Parallel processing, similar to that performed by biological stereo visual systems may be required to meet the SDI requirements.

designated impact site. After each divert of the bus, an RV is oriented for re-entry into the atmosphere, spun up for stability and deployed from the bus. To confuse and overwhelm the SDI defenses, "penetration aids" such as balloons, radar chaff, and thrusting replica decoys are deployed along with the threat RVs. An adaptive machine vision capability can provide the critical discrimination between the threat objects and the decoys which proliferate during the post boost phase.

The basis for this discrimination will be measurements gathered by passive electro-optic imaging sensors, passive IR photometric sensors, active laser radars, and other exotic systems. Passive IR imaging will not be feasible at large standoff distances, on the order of 10,000 km. However, sub-orbital trajectories which originate in the USSR and terminate in the continental U.S. are invariably sunlit over that portion of the trajectory which traverses the North Pole. Passive imaging at high resolution in the short wavelength ultraviolet light from the sun is therefore possible. For example, at a standoff distance of  $d = 10,000$  km (1.57 earth radii) and a wavelength of  $\lambda = 0.2$  micrometer, a passive interferometric imaging system with an aperture separation of  $D = 20$  meters (the collection apertures can be smaller) will provide a spatial resolution, on the target, of approximately  $(\lambda/D)(d) = 10$  cm. This resolution is adequate to determine object size and shape and to estimate the object orientation; in combination with passive multiband infrared photometry, the emissivity-temperature product of the

object may be estimated to provide a discriminant between objects of high and low thermal mass (warheads versus decoys). Additional discriminants are needed to reduce the probability of misclassification between decoys and warheads. Recent work by Sejnowski at Johns Hopkins University has demonstrated the capability of a neural network to estimate 3-D shape from shading given passive imagery. What needs to be shown next is the capability of a neural network to use this 3-D shape information for enhanced discrimination between warheads and decoys.

### 1.3 Adaptive Machine Vision for Midcourse Defense

The midcourse phase sensors will consist of pop-up sensors, derived from the probe experiment, and airborne sensors evolving from the Airborne Optical Adjunct (AOA) Experiment. The functions of these sensors include ATP, discrimination and kill assessment. Since the pop-up and airborne sensors are launched or operate over the region in the United States targeted by the Soviet missiles, a serious threat of "sensor blinding" exists as the result of fireballs in the sensor field of view. Even in the absence of nuclear detonations, extensive angular regions around the sun would be blotted out on a conventional detector array. In the event of SLBM attacks off our eastern or western coastlines, the morning and afternoon sun would play havoc with the ATP, discrimination and kill assessment functions. Simple shuttering is inadequate, because the blinding image of the sun

or of a fireball is concentrated over a limited region of the sensor field of view. What is needed is a dynamic capability to adapt to extreme variations of scene brightness, similar to the ability exhibited by the human retina. We have studied this retinal adaptation, and developed a model retina which adapts on a point-by-point basis to the scene intensity, to maintain limited visual function in the presence of dynamic blinding flashes as from a nuclear detonation.

A less extreme, but equally serious problem for an airborne sensor is the disturbed atmosphere remaining for several minutes after a nuclear detonation. Figure 4 shows the evolution of hot plasma from a one-megaton range burst occurring at a 200 km altitude, according to an unclassified Defense Nuclear Agency primer on high altitude nuclear events. At top left, the burst is shown at a early phase. At the right, after 60 seconds, the hot plasma has risen to 800 km and is highly structured. On the bottom left, after 3 minutes, the plasma continues to evolve in linear striations along the earth magnetic field, to form a severely cluttered visible and IR background for the ATP discrimination and kill assessment functions. On the bottom left of Figure 4, these striations are shown after one full hour. The striations have spread out several earth radii. During nuclear war, the long lasting contributions of precursor high altitude nuclear detonations will clearly result in an extensive disturbed and cluttered background for all types of SDI sensors... airborne, pop-up and space-based. To deal with

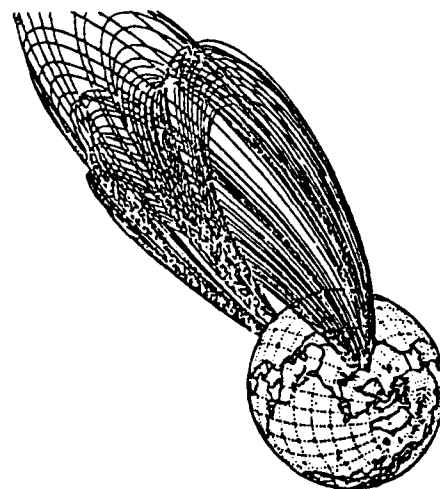
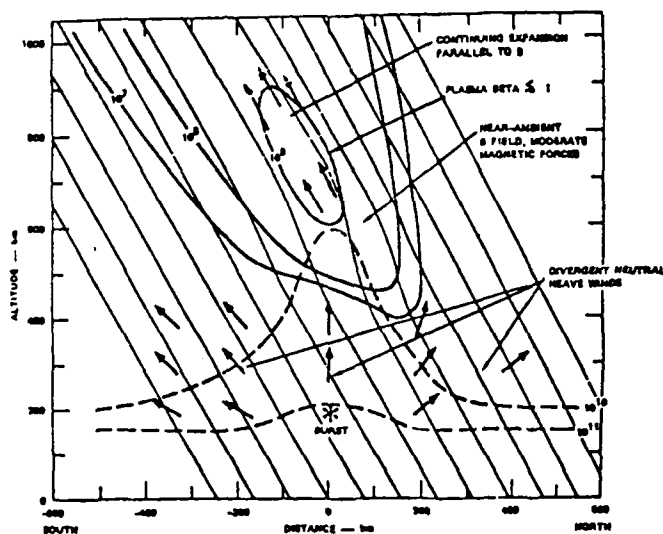
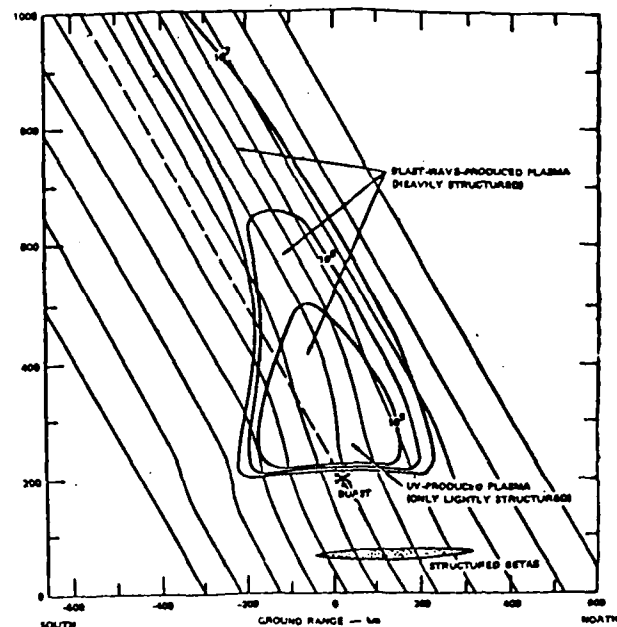
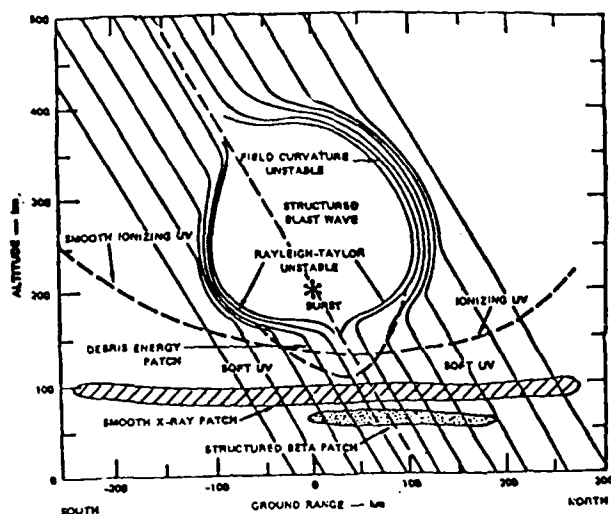


Figure 4. Top left, early phase of one-megaton range detonation at 200 km. Top right, evolution of nuclear disturbance from the blast after 60 sec. Bottom left, evolution of striated plasma along magnetic field lines after 3 min. Bottom right, striated plasma extends over several earth radii one hour after the blast. (Figures taken from unclassified DNA primer on high altitude nuclear events.)

this threat, the pattern recognition and tracking functions of these sensors must be capable of adaptive gain control (as in the model retina we have proposed) as well as powerful, clutter rejecting pattern recognition. Fusion of measurements from multiple wavebands and platforms can provide additional robustness for this function.

#### 1.4 Adaptive Machine Vision for Terminal Defense

A radar system (the Terminal Imaging Radar, TIR) is planned for the discrimination and tracking of objects leaking through to the terminal defense zone. Here the pattern classification and parallel processing capability of neural network systems to discriminate between lethal and non-lethal radar signatures (see Figure 5) is extremely promising, because the terminal defense is stressed by the large number of targets, the finite number of interceptors, and the few precious seconds remaining to commit an interceptor against a threat vehicle.

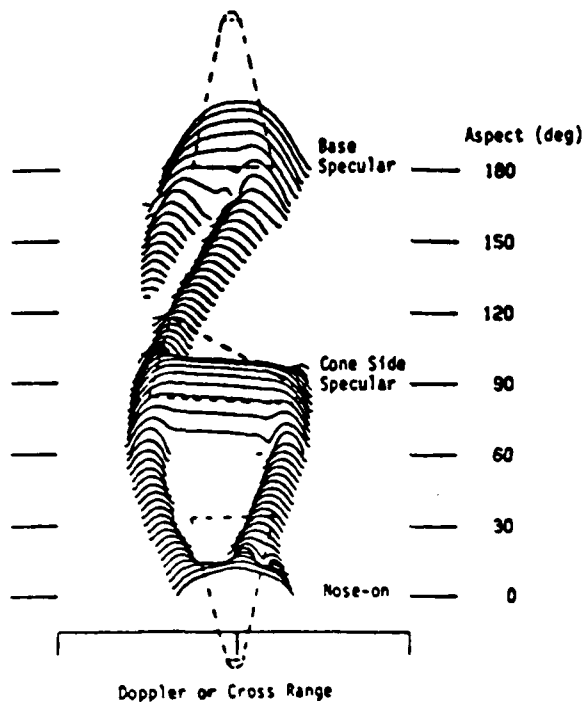


Figure 5. Simulated radar signature of an RV for aspect angles from 0° to 180°. These Doppler radar signatures may be input to a neural net for pattern classification or for fusion with EO or IR signatures to provide more robust discrimination against decoys.

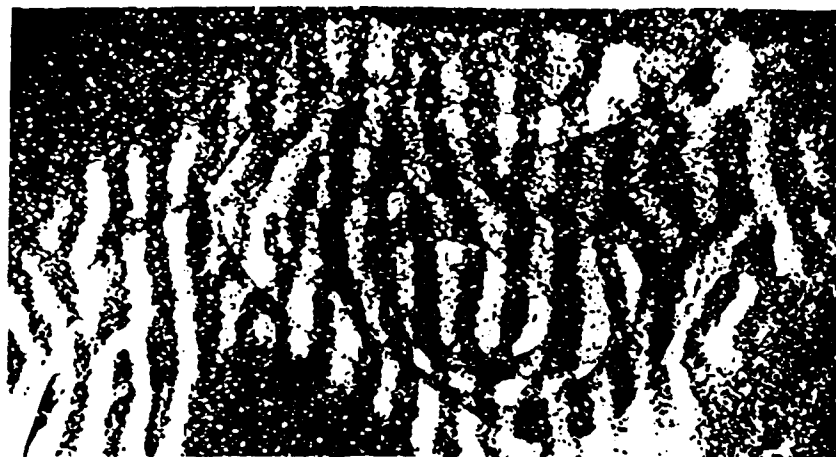


## 2.0 Application of Machine Vision to Stereo Tracking

Stereo tracking of SDI targets is attractive because it provides substantially better accuracy than tracking with a single sensor. However, the individual detected targets from the two sensors must be correctly associated before stereo tracking can begin. Since this target association problem grows as  $N^2$  when there are  $N$  targets, the computational requirements are significant. Yet this same basic function is accomplished by compact, low power biological systems.

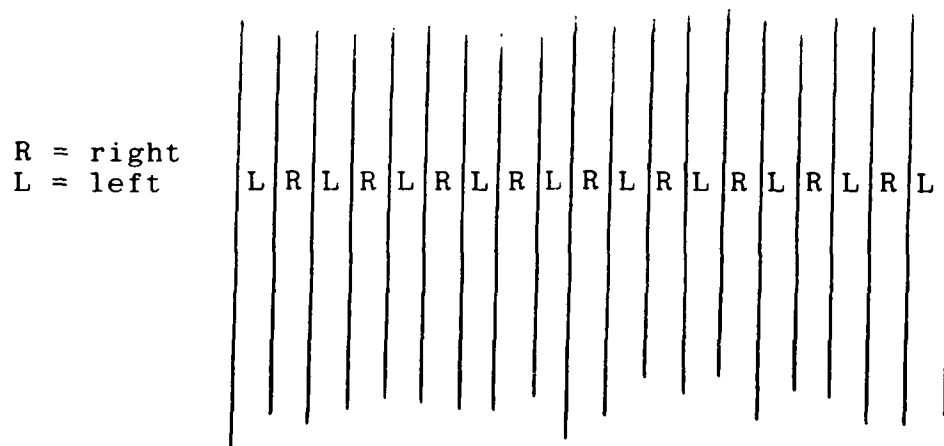
David Marr proposed two different stereo vision algorithms. We have studied these algorithms, but prefer an approach suggested by Eric Schwartz of NYU. To understand Schwartz's suggestion, it is first necessary to understand how the images captured by the right and left eyes are merged in the human visual cortex. The input from each eye is brought together in registration in the visual cortex, and organized into "ocular-dominance columns" as shown in Figure 6. The gross structure of visual cortex is a map of the entire visual field, while the microstructure contains interleaved inputs from the right and left eyes.

What advantage does this particular representation offer? Schwartz hypothesizes that this representation (formatting) of the raw data permits the extraction of depth information by means of a cepstrum-like computation.



ANATOMICAL CONFIRMATION of ocular-dominance columns came from various staining methods and from axonal-transport autoradiographs such as those shown in color on page 41. This composite autoradiograph visualizing the pattern over an area some 10 millimeters wide was made by cutting out and pasting together the regions representing layer IV in a number of parallel sections; the one in bottom illustration on page 41 and others at different depths.

Taken from "Brain Mechanisms of Vision," by D. H. Hubel and T. N. Wiesel, in *The Mind's Eye*, edited by Jeremy M. Wolfe



Idealized ocular-dominance columns

Figure 6. Top, ocular-dominance columns revealed in the visual cortex of a monkey by radioactive staining. Bottom, idealized ocular-dominance columns to be used in the cepstral stereo-tracking experiments.

His argument goes as follows: The pattern of excitation in the left ocular dominance column is a duplicate of that in the right column, except that close up objects are mapped to slightly offset columns compared with objects that are further away, as shown in Figure 7. Think of these interleaved images as an original plus its echo, where the echo offset depends upon depth.

Mathematically, the pattern represented in visual cortex may be approximated as:

$$i_r + i_l = i(x-kd, y) + i(x+kd, y) = \\ i(x, y) * \{ \delta(x-kd, y) + \delta(x+kd, y) \} ,$$

where  $i(x, y)$  is the basic image that is seen at different parallaxes by the left and right eye,  $i_r$  is the image captured by the right eye, and  $i_l$  is the image captured by the left eye. The amount of parallax depends upon the object depth  $d$ , and a scaling factor  $k$  which depends upon the lateral separation between the eyes.

To extract the depth, it is necessary to find the convolution function

$$[ \delta(x-kd, y) + \delta(x+kd, y) ] .$$

Since this convolution function is unknown, a method of blind deconvolution is required; the cepstrum is such a method.

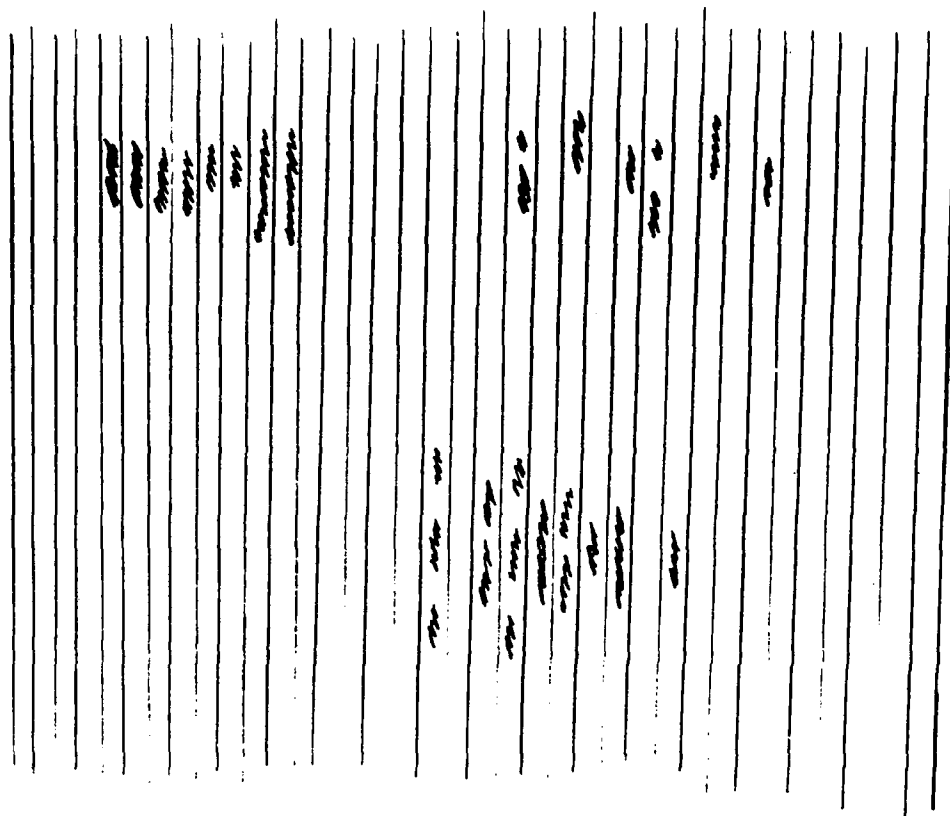


Figure 7. The depth of objects in the scene is coded in the ocular dominance representation by the offset between the corresponding image slices in the left and right ocular dominance columns. Here objects are shown at three different depths.

It is convenient to discuss the cepstrum for the general case of blind deconvolution of the function

$$h(x) = u(x) * v(u).$$

By the Fourier convolution theorem, in Fourier space we have

$$H(f) = U(f)V(f),$$

where the upper case denotes the Fourier transform has been applied to the original functions. This multiplicative form for  $H(f)$  can be reduced to a sum by taking logarithms:

$$\log(H(f)) = \log(U(f)) + \log(V(f)).$$

If the structure of  $U(f)$  is smooth and the structure of  $V(f)$  is rougher or more fine-grained, then a further Fourier transformation will separate them into distinct regions. In other words, the support of  $F(\log(U(f)))$  will not overlap with the support of  $F(\log(V(f)))$ . Here "F" denotes the Fourier transform operator. This completes Schwartz's theory for how the visual system may isolate the convolution function

$$v(x,y) = [\delta(x-kd, y) + \delta(x + kd, y)].$$

Under critical examination, the cepstrum has some difficulties, since both  $U(f)$  and  $V(f)$  must be positive valued if negative infinities and complex valued quantities are to be avoided. Of course, there is the possibility that a usable final result may be produced using log-like functions which remain finite valued for zero arguments. However, a modification of the procedure

discussed above will eliminate the occurrence of complex valued logarithms, if they really do pose a problem.

As before, use the convolution theorem to obtain

$$H(U) = U(f)$$

At this point, we take the absolute square value of  $H(f)$  before going on, so that we have

$$\log |H(f)|^2 = \log |U(f)|^2 + \log |V(f)|^2 ,$$

which avoids complex values. If we approximate the true log function with a finite valued function that saturates for very small and very large arguments, we expect that the basic properties of the log function required by the cepstrum technique will be retained, at least approximately.

After taking logarithms, instead of obtaining:

$$\log(2\cos(kdf_x)),$$

(where  $f_x$  is the Fourier space variable conjugate to  $x$ ) we now obtain

$$\log(4\cos^2(kdf_x)).$$

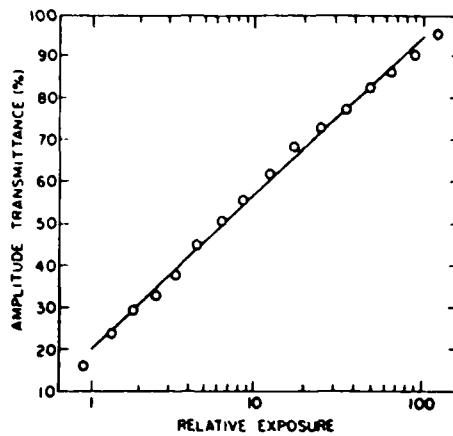
In either case, an additional Fourier transform will produce peaks whose location provides a measure of the depth  $d$ .

## 2.1 Optical Implementation

An all parallel optical implementation of the cepstral stereo fusion algorithm will require optically addressed spatial light modulators for the input, coherent optics to perform the Fourier transforms, and a logarithmic response spatial light modulator to implement the cepstrum. At the present time, spatial light modulators with a logarithmic response are not available. However, there are two ways out of this dilemma. One way is to use a thresholding spatial light modulator so that the Goodman-Kato halftone screen technique can be asked to produce a logarithmic response.

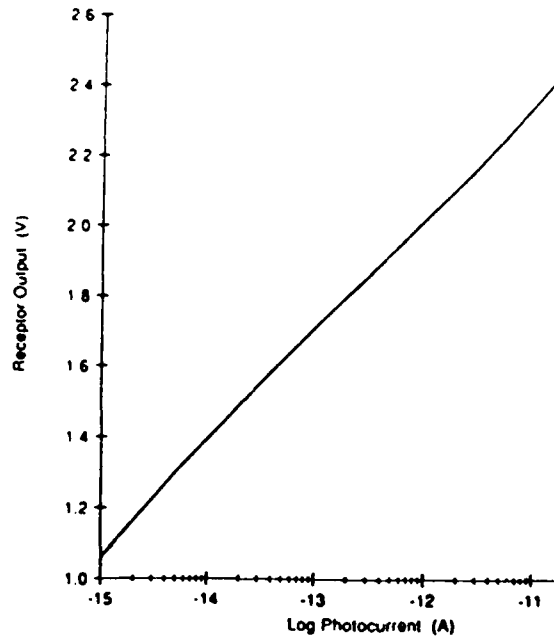
Spatial light modulators which are capable of thresholding include the variable grating mode liquid crystal device (Sawchuk and Tanguay) and a photocathode addressed lithium niobate device (Warde). See Figure 8 which shows the halftone screen technique is capable of providing a logarithmic response over two decades of input light level.

The alternative approach is to use an array of logarithmic amplifiers, each one coupled to a photoreceptor, as Carver Mead has done. His results, which are nearly logarithmic over a four decade range of input intensity, are shown also in Figure 8. It should be possible to couple such a photoreceptor array onto a liquid crystal modulator, but at the present time, such a logarithmic spatial light modulator is unavailable. If our



Measured characteristics of logarithmic halftone screen.

Goodman-Kato halftone screen results,  
Applied Optics 14, p 1817 (1975).



Measured response of logarithmic photodetector.  
Photocurrent is proportional to incident light intensity. Response  
is logarithmic over more than four orders of magnitude in in-  
tensity.

Carver Mead, CMOS logarithmic detector response  
Neural Networks, 1 p 82 (1988).

Figure 8. Logarithmic conversion results. Top, halftone screen approach,  
Bottom, CMOS photodetector approach.



demonstration using more readily available devices is encouraging, future tasks can emphasize an all parallel implementation.

The planned demonstration will use a CCD TV camera to input merged left and right views of a test scene into a liquid crystal TV display (Radio Shack) for coherent optical Fourier transformation. This transformed pattern is imaged onto a Goodman-Kato half-tone screen, and detected by a second TV camera, operated in a thresholding mode, by electronic clipping of the output video signal. The resultant signal is then applied to a second liquid crystal TV, and optically Fourier transformed. Discrete peaks in this output image correspond to the 3-D depth planes in the test scene.

The first task in our stereo tracking demonstration is to validate the concept through a digital simulation. This will provide insight into the sensitivity of the cepstral stereo algorithm to the accuracy of the logarithm function.

Figure 9 shows the laboratory set-up which provides merged and interleaved images from "right" and "left" vantage points. Only one ronchi ruling used, so misalignment problems between a pair of ronchis are eliminated. Experiments with different amounts of parallax are permitted, since the baseline between the right and left objective lenses can be varied. Finally, only a single TV camera is required to capture the input scene, so problems

with differential distortion between two cameras are eliminated.

Figure 10 shows the schematic for the cepstral processing. Coherent optics is used to perform Fourier transformation, and the logarithm is introduced through the Goodman-Kato half-tone screen, followed by clipping of the video signal in the subsequent step of image detection with a CCD TV camera.

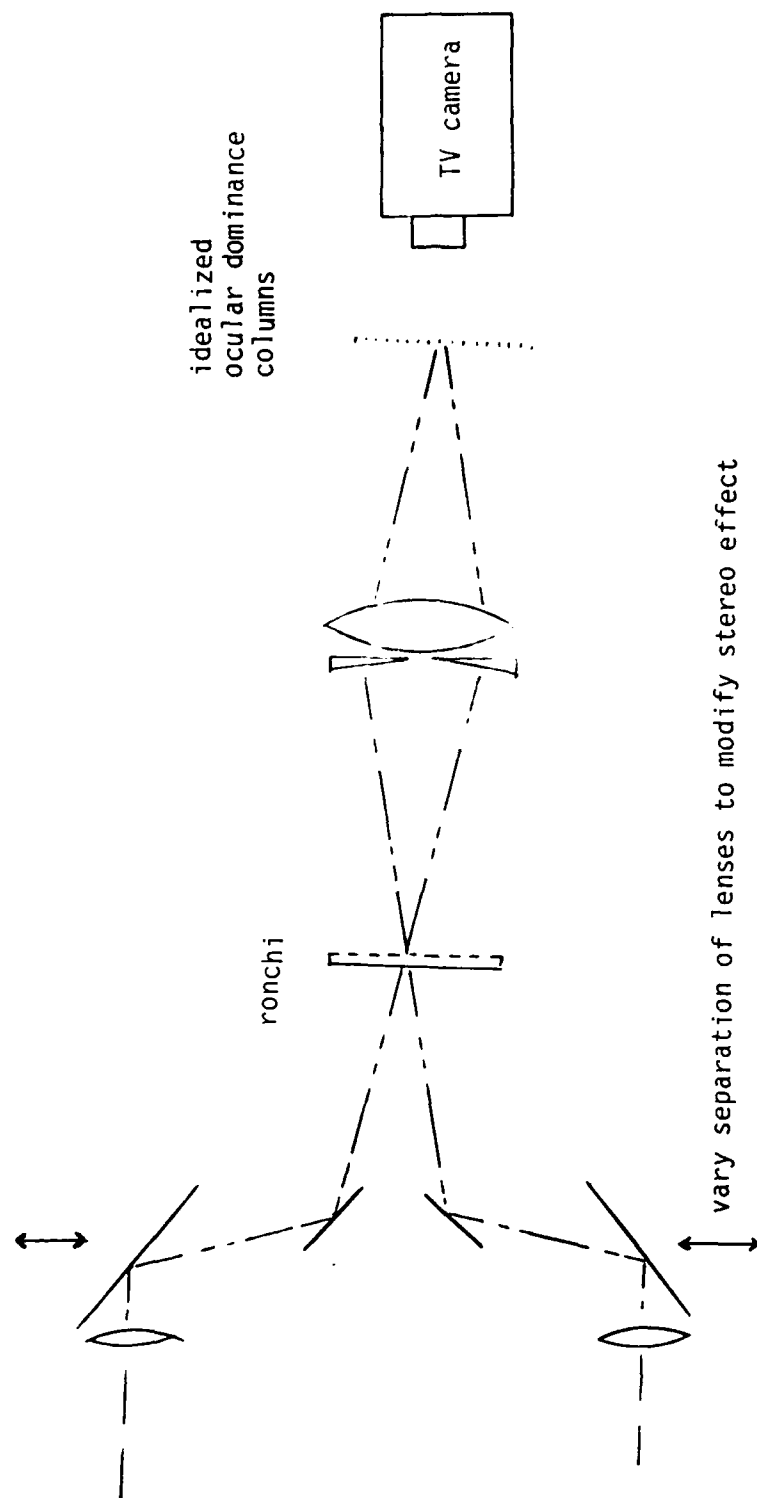


Figure 9. Laboratory set-up to generate video image in ocular-dominance column format.

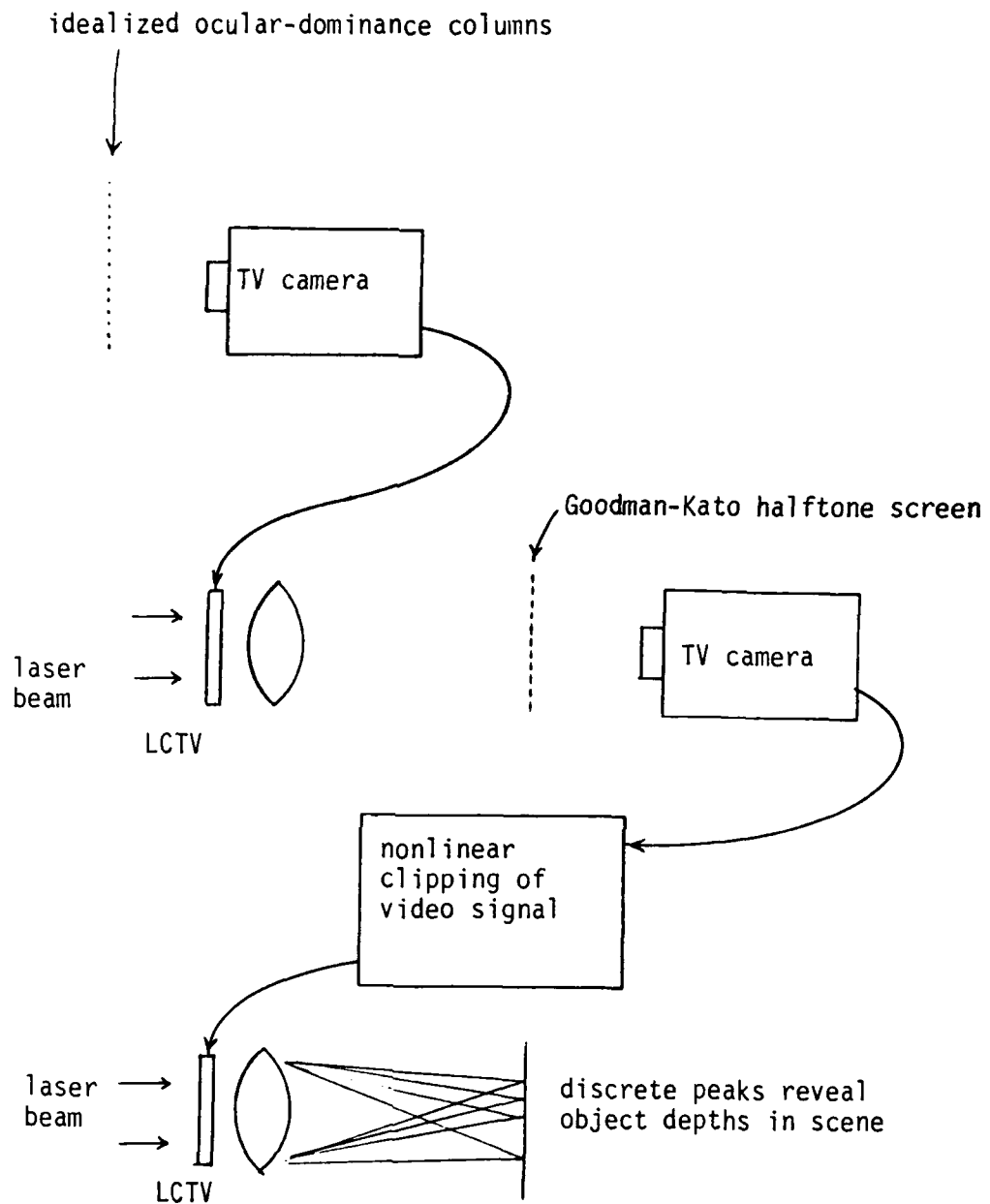


Figure 10. Block diagram of planned laboratory test of cepstral stereo tracking.

## 2.2 Digital Implementation of Stereo Algorithms

The cepstral stereo algorithm is attractive from the standpoint of optical or digital implementations. The optical implementation is uniquely capable of exploiting the computational map style of processing that is used in the interleaved input format of the cepstral algorithm. From the digital standpoint, the cepstral algorithm offers a smaller operation count than the best-known rival algorithm (the zero-crossing algorithm of David Marr and Tomaso Poggio, in "A computational theory of human stereo vision," Proc. Royal Soc. Lond. B204, pp 301-328).

In the sections that follow, we demonstrate this advantage by carefully counting the computer operations needed to carry out both algorithms. Since the speed of multiplication, addition, table look-up and algebraic sign comparison is machine dependent, such a comparison is generally made for a specific machine and the results may be different for other computers. In the present case, the dominant computation in both algorithms is an FFT, and so the algorithm ranking should be machine independent. With this in mind, we justify the comparison in Table 2.4 which counts all of the computer operations (multiply, add, look-up, sign compare) equally.

### 2.2.1 Computational Load of the Zero-Crossing Stereo Algorithm

Two distinct strategies are used by the zero-crossing algorithm. One strategy is to work with an edge-enhanced version of the stereo pair imagery. The Laplacian filter is used to perform this edge-enhancement, since this filter is not preferential to any particular orientation. The zero-crossings in the Laplacian filtered imagery are then correlated between the left and right stereo pairs. This brings us to the other strategy, which is to solve the zero-crossing association problem in stages: first at low resolution, then at double the resolution, then quadruple the resolution, and so on, up to the highest available resolution. This recursive strategy uses the low resolution depth information to guide the association of more detailed zero-crossings revealed at the next higher level of resolution.

For the remainder of this discussion, we fix the size of stereo pair images as  $N$  by  $N$ , where  $N = 2^k$ . Though not a requirement of the algorithm, it is helpful in analyzing the computational load, since the recursive processing uses image resolutions which change by a factor of 2 between stages. To prepare the blurred imagery needed for a given stage, we merely need to replace every 2 by 2 square of pixels in the previous image with a single pixel. This involves summing up the 4 pixel values at each of the new sample locations. The original  $N$  by  $N$  image goes over to an  $N/2$  by  $N/2$  image, and the computation count is:

$$(3 \text{ additions per pixel})(N^2/4 \text{ pixels}) = (3/4)N^2 \text{ additions.}$$

See Table 2.1 where this result is tabulated. The blurring is repeated in stages to produce stereo pair images sized  $N/4$  by  $N/4$ ,  $N/8$  by  $N/8$ , etc. At each successive stage, the number of pixels decreases a factor of 4, so the computation count over all stages of blurring forms a geometric series:

$$(3/4)N^2 [1 + 1/4 + 1/16 + \dots] \text{ additions.}$$

The infinite geometric series sums to  $4/3$ , and since the series is rapidly convergent, the truncated sum may be approximated by  $4/3$ .

So the computation count for blurring totals  $2N^2$  additions. The factor of 2 comes from the left and right stereo pairs. See Table 2.1.

In the next step of the zero-crossing algorithm, the Laplacian filter is applied to the blurred images and the resultant zero-crossings are located. The Laplacian filter combines each pixel with the weighted sum of its 4 nearest neighbors. There are 4 additions and 1 multiplication for each filtered value. Hence, to filter an  $N$  by  $N$  image requires:

$$4N^2 \text{ additions} + N^2 \text{ multiplications.}$$

See Table 2.1 where this result is tabulated. Once again, a factor of  $4/3$  is applied to include the filtering for all resolution scales, and a factor of 2 for both left and right

Table 2.1 COMPUTATION COUNT FOR  
ZERO-CROSSING ALGORITHM

KEY:	ADDITIONS.....[+]	A FACTOR OF 2.....x2
	MULTIPLICATIONS.....[*]	A FACTOR OF 4/3.....x4/3
	SIGN CHECKS.....[sgn]	A FACTOR OF (N/m)*(N/m).....x(N/m)^2
	N*N.....N^2	BASE 2 LOGARITHM.....log

	COMPUTATION COUNTS:				SUBTOTALS:		
	OVER m x m OR M x M ARRAY	OVER (N/m)^2 NEIGHBOR- HOODS	OVER ALL SCALES	BOTH STEREO PAIRS	ADDS [+]	MULTIPLIES [*]	SIGN CHECKS [sgn]
FORMING BLURRED IMAGES	$3/4 N^2[+]$	n/a	$x4/3$	x2	$2N^2$		
LAPLACIAN FILTERING	$4N^2[+]$ $N^2[*]$	n/a	$x4/3$	x2	$32/3N^2$	$8/3N^2$	
ZERO- CROSSING LOCATION	$2N(N-1)[sgn]$	n/a	$x4/3$	x2			$(16/3)N(N-1)$
CROSS- CORRELATION							
DIRECT:	$m^2(m-1)[+]$	$x(N/m)^2$	$x4/3$	x1	$4/3N^2(m-1)$		
	$m^3[*]$					$4/3N^2(m)$	
- OR -							
IN FOURIER- SPACE:							
FORWARD FFT	$3m^2 \log(2m)[+]$ $2m^2 \log(2m)[*]$	$x(N/m)^2$	$x4/3$	x2	$8N^2 \log(2m)$	$16/3N^2 \log(2m)$	
PRODUCT OF TRANSFORMS	$4m^2[+]$ $8m^2[*]$	$x(N/m)^2$	$x4/3$	x1	$16/3N^2$	$32/3N^2$	
INVERSE FFT	$6m^2 \log(2m)[+]$ $4m^2 \log(2m)[*]$	$x(N/m)^2$	$x4/3$	x1	$8N^2 \log(2m)$	$16/3N^2 \log(2m)$	



stereo pairs, so the computation count for Laplacian filtering totals:

$$(32/3)N^2 \text{ additions} + (8/3)N^2 \text{ multiplications.}$$

The search for zero-crossings involves a sign comparison between each pixel and its nearest neighbors. In an  $N$  by  $N$  array there are  $2N(N-1)$  places for the pixels to change from positive to negative or vice-versa. Including the factor of  $4/3$  for all resolution scales, and the factor of 2 for both left and right stereo pairs, the computation count for locating zero-crossings totals:

$$(16/3)N(N-1) \text{ sign comparisons.}$$

See Table 2.1 where this result is tabulated.

The final step of the zero-crossing algorithm is the association of zero-crossings between the left and right stereo pairs. The associations are made on the basis of peaks in the cross correlation between small neighborhoods taken from the left and right stereo pairs. The cross-correlation may be made directly, or they may be carried out via Fourier space, using the convolution theorem. The computation count results are different, depending upon the correlation method. Despite the well-known computational efficiency of the Fast Fourier Transform (FFT) for large transforms, direct cross-correlation computation may actually be more efficient if the correlations

are made over small neighborhoods. Take the neighborhood size as  $m$  by  $m$ . The cross-correlation is carried out only along the axis separating the stereo pairs, and is performed for lags of  $\pm m/2$ . Direct correlation requires  $m$  multiplications and  $m-1$  adds for each of the  $m$  lags, or  $m^3$  multiplications and  $m^2(m-1)$  additions for each  $m$  by  $m$  neighborhood. An  $N$  by  $N$  image contains  $(N/m)^2$  such neighborhoods, and the cross correlation over all resolution scales increases the operation count by a factor of  $4/3$ . The total operation count for each cross-correlation tabulated in Table 2.1 is therefore:

$$(4/3)N^2m \text{ multiplications} + (4/3)N^2(m-1) \text{ additions.}$$

The alternative correlation approach uses the FFT to Fourier transform corresponding neighborhood for each stereo pair, and then inverse transforms the product of these transforms to obtain the cross-correlation. Since the cross-correlation is performed along only one axis, a 1-D transform along the axis separating the stereo pairs is required for each strip of pixels.

The cross-correlation between strips from each stereo pair is to be carried out over lags of  $\pm m/2$  pixels. One of the strips must be  $2m$  pixel long, the other  $m$  pixels long. The  $m$  pixel strip must be augmented with  $m$  additional zero valued samples (zero padding) to keep both the transforms the same length,  $2m$ .

An FFT of length  $K$  requires  $K/2 \log_2(K)$  complex multiplications and  $K \log_2(K)$  complex additions. However, our input data is real-valued, and the FFT algorithm also computes the transform for the (all zero) imaginary part of the input. For real-valued inputs, it is possible to reduce the computations in half. The FFT of real-valued data of length  $2m$  samples then requires:

$3m \log_2(2m)$  additions and  $2m \log_2(2m)$  multiplications.

Each  $m$  by  $m$  neighborhood requires  $m$  such transforms, and there are  $(N/m)^2$  neighborhoods in an  $N$  by  $N$  image. The total operation count for the forward Fourier transforms including the 2 stereo pairs and all resolution scales is:

$8N^2 \log_2(2m)$  additions and  $(16/3)N^2 \log_2(2m)$  multiplications.

The next step is the multiplication of the transformed data from the left and right stereo pairs. With the zero padding, the length of each transformed strip is  $2m$ , and each neighborhood contains  $m$  such steps, so there are  $2m^2$  multiplications for each  $m$  by  $m$  neighborhood. Counting all of the  $(N/m)^2$  neighborhoods in each  $N$  by  $N$  image, and including a factor of  $4/3$  for all of the resolution scales, the total operation count listed in Table 2.1 for multiplying in Fourier space comes to:

$(8/3)N^2$  complex multiplications, or

$(16/3)N^2$  additions and  $(32/3)N^2$  multiplications.

The final inverse transform step is full-complex and has twice

the operation count as the individual forward transforms (but there no longer are 2 stereo pairs) so Table 2.1 includes inverse transform operations totalling:

$8N^2 \log_2(2m)$  additions and  $(16/3)N^2 \log_2(2m)$  multiplications.

Table 2.2 uses a spread-sheet format to compare the operation count for direct cross-correlation with that for correlation via Fourier space. Direct cross-correlation is most efficient for  $m \leq 64$ . However, for high resolution in depth, larger values of  $m$  will be needed.

#### 2.2.2 Computational Load of the Cepstral Stereo Algorithm

The operations of the cepstral stereo algorithm are applied over  $(N/m)^2$  neighborhoods each of size  $2m$  by  $m$ . Zero-padding by  $2m$  zeros along the axis oriented from one stereo pair to the other brings the Fast Fourier Transform (FFT) length up to  $4m$ . The operation count for each (all-real) transform of length  $4m$  is:

$6m \log_2(4m)$  additions and  $4m \log_2(4m)$  multiplications.

For each  $2m$  by  $m$  neighborhood, there are  $m$  of these transforms, and there are  $(N/m)^2$  neighborhoods so the total operation count for the first set of transforms is:

$6N^2 \log_2(4m)$  additions and  $4N^2 \log_2$  multiplications.

The next step is to form the absolute value squared of each

Table 2.2 Zero-Crossing Stereo Algorithm: Direct vs Fourier-Space Correlation

OPERATIONS PER PIXEL WITH DIRECT CROSS-CORRELATION

	ADDITIONS	MULTIPLICATIONS	SIGN COMPARISONS
FORMING BLURRED IMAGES.....	2.00	0.00	0.00
LAPLACIAN FILTERING.....	10.67	2.67	0.00
ZERO-CROSSING LOCATION.....	0.00	0.00	5.33
SUBTOTAL	12.67	2.67	5.33

SUBTOTAL + OPERATION COUNT FOR CROSS-CORRELATION

CROSS-CORRELATION IS A FUNCTION OF NEIGHBORHOOD SIZE:

NEIGHBORHOOD SIZE (n)	BASE 2 LOG OF n	ADDITIONS	MULTIPLICATIONS	SIGN COMPARISONS	TOTAL OPERATIONS PER PIXEL AS A FUNCTION OF NEIGHBORHOOD SIZE (n) (DIRECT CROSS-CORRELATION)
6.00	3.00	22.00	13.33	5.33	40.67
16.00	4.00	32.67	24.00	5.33	62.00
32.00	5.00	54.00	45.33	5.33	104.67
64.00	6.00	98.67	86.00	5.33	190.00
128.00	7.00	182.00	173.33	5.33	360.67
256.00	8.00	352.67	344.00	5.33	702.00
512.00	9.00	694.00	685.33	5.33	1384.67
1024.00	10.00	1376.67	1368.00	5.33	2750.00
2048.00	11.00	2742.00	2733.33	5.33	5480.67
4096.00	12.00	5472.67	5464.00	5.33	10942.00
8192.00	13.00	10934.00	10925.33	5.33	21864.67
16384.00	14.00	21856.67	21848.00	5.33	43710.00

OPERATIONS PER PIXEL WITH CROSS-CORRELATION IN FOURIER-SPACE

	ADDITIONS	MULTIPLICATIONS	SIGN COMPARISONS
FORMING BLURRED IMAGES.....	2.00	0.00	0.00
LAPLACIAN FILTERING.....	10.67	2.67	0.00
ZERO-CROSSING LOCATION.....	0.00	0.00	5.33
SUBTOTAL	12.67	2.67	5.33

SUBTOTAL + OPERATION COUNT FOR CROSS-CORRELATION

CROSS-CORRELATION IS A FUNCTION OF NEIGHBORHOOD SIZE:

NEIGHBORHOOD SIZE (n)	BASE 2 LOG OF n	ADDITIONS	MULTIPLICATIONS	SIGN COMPARISONS	TOTAL OPERATIONS PER PIXEL AS A FUNCTION OF NEIGHBORHOOD SIZE (n) (CROSS-CORRELATION IN FOURIER-SPACE)
6.00	3.00	62.00	56.00	5.33	143.33
16.00	4.00	96.00	66.67	5.33	170.00
32.00	5.00	114.00	77.33	5.33	196.67
64.00	6.00	130.00	86.00	5.33	223.33
128.00	7.00	146.00	98.67	5.33	250.00
256.00	8.00	162.00	109.33	5.33	276.67
512.00	9.00	178.00	120.00	5.33	303.33
1024.00	10.00	194.00	130.67	5.33	330.00
2048.00	11.00	210.00	141.33	5.33	356.67
4096.00	12.00	226.00	152.00	5.33	383.33
8192.00	13.00	242.00	162.67	5.33	410.00
16384.00	14.00	258.00	173.33	5.33	436.67

transform value. This involves  $4m^2$  additions and  $8m^2$  multiplications over each neighborhood, and over the  $(N/m)^2$  neighborhoods totals:

$4N^2$  additions and  $8N^2$  multiplications.

Taking the logarithm of each value requires a table look-up and an interpolation. The input value must first be converted through some bit-level manipulations into a table address and a residual value,  $x$ , for the interpolation function. The log function is convex, and so quadratic interpolation should be effective. Evaluation of a quadratic in  $x$  requires 2 additions and 2 multiplications when expressed in the form:

$$x(Ax+B)+C.$$

The quantities  $A$ ,  $B$  and  $C$  are obtained from the look-up table. Each neighborhood contains  $4m^2$  values to be converted into logarithms, and over  $(N/m)^2$  neighborhoods the total operation count comes to:

$8N^2$  additions,  $8N^2$  multiplications, and  $4N^2$  table look-ups.

The final step in the cepstral stereo algorithm is an inverse Fourier transformation. The operation count is the same as for the initial set of forward transform,

$6N^2 \log_2(4m)$  additions and  $4N^2 \log_2(4m)$  multiplications.

These results are listed in Table 2.3.

Table 2.3 COMPUTATIONAL COUNT FOR THE  
CEPSTRAL STEREO ALGORITHM

KEY: ADDITIONS.....[+]  
MULTIPLICATIONS.....[\*]  
TABLE LOOK-UPS.....[tlu]  
N\*N.....N<sup>2</sup>

A FACTOR OF 2.....x2  
A FACTOR OF (N/m)\*(N/m)....x(n/m)<sup>2</sup>  
BASE 2 LOGARITHM.....log

	COMPUTATION COUNTS:			SUBTOTALS:		
	OVER SINGLE STRIP (2m pixels + 2m zero-pad)	OVER NEIGHBORHOOD (m STRIPS)	OVER (N/m) <sup>2</sup> NEIGHBORHOODS	ADDS [+]	MULTIPLIES [*]	TABLE LOOK-UPS [tlu]
FORWARD FFT	6mlog(4m)[+] 4mlog(4m)[*]	xm	x(N/m) <sup>2</sup>	6N <sup>2</sup> log(4m)	4N <sup>2</sup> log(4m)	
ABSOLUTE VALUE SQUARED	4m[+] 8m[*]	xm	x(N/m) <sup>2</sup>	4N <sup>2</sup>	8N <sup>2</sup>	
BASE 2 LOG EVALUATION	8m[+] 8m[*] 4m[tlu]	xm	x(N/m) <sup>2</sup>	8N <sup>2</sup>	8N <sup>2</sup>	4N <sup>2</sup>
INVERSE FFT	6mlog(4m)[+] 4mlog(4m)[*]	xm	x(N/m) <sup>2</sup>	6N <sup>2</sup> log(4m)	4N <sup>2</sup> log(4m)	

### 2.3 Stereo Algorithm Comparison

Table 2.4 compares the computational efficiency of the cepstral and the zero-crossing stereo algorithms. The number of operations per image pixel depends upon the size of the neighborhoods which are processed to extract depth information. Precise depth measurements will require large neighborhood sizes. The Table shows that the zero-crossing algorithm is most efficient for neighborhood sizes of 64x64 or smaller. For larger neighborhoods, the cepstral algorithm is more efficient, because its computational count grows as  $20\log_2(m)$  versus  $(80/3)\log_2(m)$  for the zero-crossing algorithm.



Table 2.4 Computational Efficiency: Cepstral vs Zero-Crossing

NEIGHBORHOOD SIZE m	BASE 2 LOG OF m	CEPSTRAL	ZERO-CROSSING
		TOTAL OPERATIONS PER PIXEL	TOTAL OPERATIONS PER PIXEL
8.00	3.00	132.00	40.67
16.00	4.00	152.00	62.00
32.00	5.00	172.00	104.67
64.00	6.00	192.00	190.00
128.00	7.00	212.00	250.00
256.00	8.00	232.00	276.67
512.00	9.00	252.00	303.33
1024.00	10.00	272.00	330.00
2048.00	11.00	292.00	356.67
4096.00	12.00	312.00	383.33
8192.00	13.00	332.00	410.00
16384.00	14.00	352.00	436.67

### 3.0 Neocognitron Evaluation

The Neocognitron is an adaptive neural network for pattern recognition developed by Kunihiro Fukushima and his colleagues. (See K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," Biol. Cybernetics, Vol. 36 pp 193-202, 1980). The Neocognitron has a pattern recognition capability that is tolerant to scale changes, distortion and pattern registration (shift). Training of the Neocognitron is accomplished by an unsupervised learning procedure. It is not necessary to instruct the network; instead, a competitive learning algorithm creates new exemplars as novel patterns are introduced during the learning phase. Although a teacher is not utilized to correct the network errors, careful pattern selection and repetitive pattern presentation is essential to successful learning. In this sense, the "invisible hand" of the teacher is part of the learning process. Unlike a human infant, the Neocognitron cannot make sense out of unedited confusion.

The shift, scale and distortion tolerance of the Neocognitron is accomplished gradually over a series of feedforward processing stages. The general architecture is illustrated in Figure 11, which is taken from a paper by William Stoner and Terry M. Schilke, "Pattern Recognition with a Neural Net," SPIE Vol 698, Real Time Signal Processing IX pp 170-181 (1986).

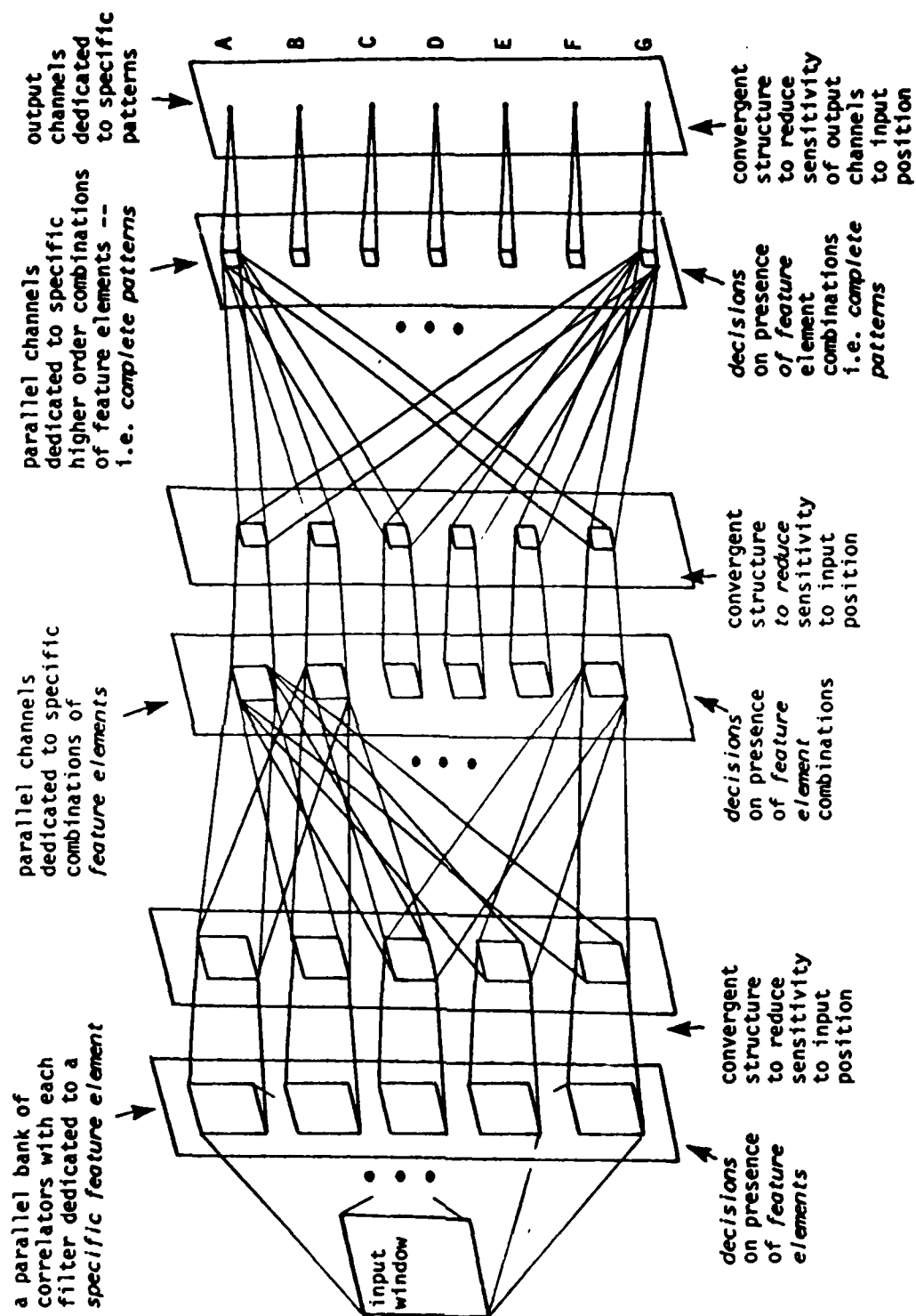


Figure 11. Hierarchical, feedforward architecture for pattern recognition.

The "front-end" of the Neocognitron consists of a bank of parallel correlators, which become dedicated during training to specific feature elements. The outputs of these correlators are nonlinear in the sense that only "above average" correlations are past on to the next stage. So even in the first stage, decisions are made which reduce the information flow in an attempt to select the significant features of the input pattern. After feature extraction, the spatial resolution is slightly degraded in order to introduce tolerance to shift, scale and distortion. This sequence of nonlinear feature extraction followed by degradation of spatial resolution is then repeated in order to integrate (fuse) feature element information that is spread out over the input image. For example, to recognize a rose, the separate features for stem, thorn, leaves, petals, etc. must be combined. Since these feature elements occur at different places in the input image, it is necessary to bring them together by reducing the spatial resolution, so that higher order correlators in subsequent stages can detect combinations of feature elements such as (thorn + stem). After several stages of feature detection and integration, complete patterns are detected, with insensitivity to the location, scale, and distortion of the pattern.

In the 1986 paper by Stoner and Schilke cited above, a model Neocognitron was implemented on an IBM PC AT. The input image was limited to a 14x14 array, because of the modest computing power of the PC AT. A typical application of the Neocognitron

might require a 128x128 input array, and processing times a 1000x faster than the processing time on the PC AT (hours). A fast mainframe, or a dedicated parallel machine would help, but how much? In order to predict the processing speed of a larger Neocognitron model on a faster machine, it is necessary to understand how the operation count of the Neocognitron grows as the scale of the input array increases. This is analyzed in Section 3.2.

In advance of this analysis, we are prompted by the sluggish speed of the Neocognitron on the IBM PC AT to ask if there are alternative candidates for SDI target discrimination missions requiring tolerance to shift, scale, and distortion.

### 3.10 Pattern Recognition with Tolerance to Shift, Scale and Aspect Angle

We now pose an important question: what is a good strategy for achieving pattern recognition independent of shift, scale, and aspect angle? To address this question, we first characterize the problem in mathematical form. The following discussion shows that the image changes resulting from positional shifts, scale variations and aspect changes may be accommodated by adding 6 extra degrees of freedom to the target image. By searching over a range of parameters representing the additional degrees of freedom, the target image may be matched, even in the presence

of a scale or aspect change.

Consider first the trivial parametric representation of shifts. Shifts may be parameterized in terms of the horizontal and vertical components of shift,  $x_s$  and  $y_s$ . In an ideal situation, suppose the coordinates overlaying the target are  $(x, y)$ , with origin at the centroid of the target. If the target is shifted by  $(x_s, y_s)$  from the ideal position, the coordinates overlaying the target are given by

$$x' = x + x_s \quad \text{and} \quad y' = y + y_s.$$

Similarly, scale changes by a scale magnification factor of  $M$  are represented by

$$x' = Mx \quad \text{and} \quad y' = My, \quad \text{or}$$

$$x' = Mx + x_s \quad \text{and} \quad y' = My + y_s,$$

in the presence of a shift. If the shift is removed, and the coordinates are distorted logarithmically, a scale change is converted into a shift

$$\log(x' - x_s) = \log(x) + \log(M).$$

This transformation does not eliminate the scale parameter  $M$ . The parameter space that must be searched to find a match with the nominal target image is just as large as before. The equivalence between a shift by  $\log(M)$  and a scale change by a factor of  $M$  illustrates that the dimension of the search space

is more significant than the form of the parameterization.

As the aspect angle of an object changes, most of the change in the image is a continuous function of the angle change. However, when the changing aspect reveals or conceals facets of the object, the image changes discontinuously in the first derivative. In mathematical language, the image is a piecewise continuous function of aspect angle. Many objects have a limited number of facets, and their appearance may be captured with a few well selected views. This is especially true of objects which are aerodynamically streamlined. In such cases, it is feasible to represent the object image with a reasonable number,  $P$ , of discrete models, which jointly comprise a piecewise continuous image model, valid for all aspect angles. The  $j$ th of these models may be based upon a continuous distortion  $(x, y) \rightarrow (x', y')$  of the image  $I_j$  corresponding to the  $j$ th aspect of the object:

$$I_j(x, y) = I_j(x', y').$$

The coordinates  $x', y'$  are distorted from  $x, y$  as a function of incremental changes in aspect angle about the  $j$ th aspect. Assume  $(x', y')$  is expressed as a polynomial in  $(x, y)$ :

$$x' = \sum_{i=0}^D \sum_{q=0}^i a_{i,q} x^q y^{i-q}, \quad y' = \sum_{i=0}^D \sum_{q=0}^i b_{i,q} x^q y^{i-q}$$

The degree of the polynomials need be no higher than  $D=1$ , and the coefficients  $a_{i,q}$  and  $b_{i,q}$  will be functions of the three

parameters which define the deviation in object aspect angle from the  $j$ th aspect: pitch, roll, and yaw.

In summary, shift tolerant pattern recognition requires searches over a 2 parameter space. Scale variations may be accomodated with one more parameter, and aspect angle changes with 3 additional parameters. The most general capability of shift, scale, and aspect tolerant pattern recognition requires a search over a 6 dimensional space.

### 3.11 Alternative Approaches to Invariant Pattern Recognition

By virtue of the shift-invariant connection patterns within slabs, and the gradual, module-by-module elimination of spatial resolution, the Neocognitron is hard-wired for shift, distortion and scale tolerant pattern recognition. Is there a better way to achieve these properties? For example, why not use back-propagation to train a neural net for these properties? Or, why not use the shift-invariant property of the Fourier power spectrum to obtain shift invariance?

It is possible to demonstrate analytically the utter impossibility of training for shift invariance, distortion and scale using a learning algorithm like backpropagation. The difficulty arises from the astronomically large number of cases on which the neural net must be trained in order to adequately



sample possible instances of shifted patterns. Similar arguments can be developed for scale and distortion invariance, because they too require a search over a low dimensional parameter space.

To keep the argument clear, we restrict the analysis to binary input patterns. That is, pixels are either white or black. An  $N \times N$  input array then admits  $2^{N \times N}$  possible patterns. For a reasonable array size, say  $128 \times 128$ , this is an enormous number,  $2^{16384}$ . Assume that the array has a doughnut topology so that as a pattern is shifted that portion which falls off the edge re-appears on the opposite side. Then it is clear that a given pattern can be re-located to any one of the  $N^2$  array locations. If all of these  $N^2$  shifted versions of a pattern are classified as equivalent to one another, there are  $2^{N \times N} / N^2$  classes of patterns, disregarding shifts. This is still an enormous number; for  $N=128$  we have  $2^{17384} / 2^{14} = 2^{16370}$ .

To train for shift invariance, the training set should adequately sample the space of  $2^{16370}$  pattern classes. To put this in perspective, there are less than  $2^{25}$  seconds in a year, and so the naive idea of training a network for shift-invariance just isn't possible, because it would take too long. It appears that training for distortion or scale invariance is fraught with the same problem. We conclude that building in shift, distortion, and scale invariance is the proper approach.

However, it is not clear that the Neocognitron architecture is the best approach, because alternative approaches such as using the Fourier power spectrum or invariant moments can provide some of the desired invariances. Simulations are required to rank the effectiveness of these alternative approaches under realistic conditions; for example, the performance of power spectrum and invariant moment approaches is degraded in the presence of background clutter. These known deficiencies must be quantified and compared to the Neocognitron performance. It is also necessary to discover how the Neocognitron architecture grows with the size of the input image, so that the speed of the Neocognitron may be projected for large input images.

### 3.2 Computational Load of the Neocognitron

In the following discussion, we determine the scaling law and computational count for one complete cycle of Neocognitron operation. To do this, it is necessary to introduce some specialized terminology: module, layer, slab, channel and unit.

Each processing stage is accomplished in a module. The modules are numbered from  $l=1$  to  $l=L$  where  $L$  is the last module. The modules consist of two processing layers, the  $S$  layer which performs correlations and the  $C$  layer which reduces the spatial resolution. Within each layer are multiple slabs. The distinct slabs comprise separate channels through the architecture.

Within a slab, all of the connections are shift-invariant, but different slabs have distinct connection patterns. Within the  $l$ th module, slabs are numbered from 1 to  $K_1$ , where  $K_1$  is the number of distinct channels through the module. Within each slab are individual processing units, which perform the nonlinear summation of activity within their receptive field (weighted connection pattern).

### 3.2.1 Scaling of the Neocognitron

The Neocognitron can be considered a multidimensional system. The input array is  $N$  by  $N$ ; there are  $L$  cascaded modules in the (feedforward) processing chain; and within each module there are multiple processing channels  $K_1$  which terminate in the  $K_L$  output classes. How do these parameters scale as the input array size is varied? What is the limitation on the number of output channels? Finally, how does the computation count scale with  $N$  and  $K_L$ ?

We consider first the scaling of  $L$  with  $N$ .  $L$  is independent of the number of output channels  $K_L$ . As  $K_L$  increases the number of processing modules remains the same, but each module needs additional channels to feed the increased number of output channels.

The increase of  $L$  with  $N$  is a consequence of the shift-invariance property of the Neocognitron. As an input pattern is shifted, the output channels must remain fully connected to the pattern of input activity. At the same time, any motion of an input pattern must be reduced in each module, so that at the last module, there is complete insensitivity to shifts at the input. This property of the Neocognitron is achieved gradually. Each module provides a little more shift insensitivity by virtue of the reduction in slab size between the  $S$  and  $C$  layers in the module. The receptive fields of adjacent units on a  $C$  slab are made to overlap on the preceding  $S$  slab so that as activity moves around in the  $S$  slabs in accord with shifts of the input pattern, the activity registered in the  $C$  slab also moves around slightly, but the activity level is nearly constant. This goal is satisfied in an integrated sense if the receptive fields fall off linearly to zero with the distance from the center of the field, and the adjacent receptive fields overlap 50%.

If the  $rxr$  receptive fields in the  $C$  slab did not have to overlap, then an  $nxn$   $S$  slab could be covered by an  $(n/r) \times (n/r)$   $C$  slab. The requirement for a 50% overlap doubles the number of sample points along each dimension of the  $C$  slab. Including edge effects, a

$$(2n/r + 1) \times (2n/r + 1)$$

$C$  array is needed for an  $nxn$   $S$  array. In the following, edge

effects are ignored, so we take the ratio of the C slab array size to the preceding S slab array size as  $(2/r)^2$ .

The point is that the array sizes decrease geometrically from module to module. Since the last module has only one pixel in each channel, we have that the number of modules,  $L$  must satisfy

$$N(2/r)^L \leq 1, \text{ or } L \geq \log_2(N)/\log_2(r/2).$$

This criterion fixes  $L$ , number of modules, when  $N$  and  $r$  are given. The receptive field size need not be constant from module to module. However we assume a single value for  $r$  in order to make the analysis clear. The receptive field sizes of the S layer should be optimized to the size of the smallest features in the preceding C layer. If they are bigger than this, scale and distortion tolerance will be reduced. To preserve information in passing from the S layer to the C layer within each module, the receptive fields in the C layer should not be significantly larger than the receptive fields in the preceding C and S layer mapping. In the demonstration we made on the IBM PC AT, the receptive field size was 5x5, (see the above cited paper by Stoner and Schilke).

### 3.2.2 Operation Count for One Neocognitron Cycle

To organize the operation count, we split the Neocognitron computations into two classes. The primary class consists of multiply-accumulate computations which generate the excitatory and inhibitory inputs to individual S and C units. This class of operations may be taken into account by counting the connections between layers. This is done in Sections 3.2.2.1 and 3.2.2.2. The excitatory and inhibitory inputs to an individual unit are combined by a nonlinear function to generate the unit output. The operation count for nonlinear combining is facilitated by counting the number of S and C units. This is done in Sections 3.2.2.3 and 3.2.2.4. The results are analyzed in Section 3.2.2.5.

#### 3.2.2.1 Connections Between Modules

Consider the connections between an S slab and the  $N \times N$  input array. The output of each unit in the S slab is computed by a nonlinear summation over the  $r^2$  connections of the  $r \times r$  receptive field in the input array. The S slabs in the first module have the full  $N \times N$  array size, and counting both excitatory and inhibitory inputs to the S units, the total number of connections in the S slab to the input array is  $2N^2r^2$ . The first module has several S slabs, numbered from 1 to  $K_1$ ;

therefore there are  $2K_1 N^2 r^2$  connections from the input array to the first S layer.

In the second module, the S slabs are connected to C slabs of dimension  $N(2/r) \times N(2/r)$ . So connections from any given S slab in the second module to any given C slab in the first module are  $2N^2(2/r)^2 r^2$  in number. There are multiple channels in both the first ( $K_1$ ) and second ( $K_2$ ) modules, so in all there are

$$2K_1 K_2 N^2 (2/r)^2 r^2$$

connections between the first and second modules. This pattern continues from module to module, and in general the number of connections from the  $l$ th to the  $(l + 1)$ th module is

$$2K_1 K_{l+1} N^2 (2/r)^{2l} r^2.$$

If we define  $K_0 = 1$ , this general expression also applies to the connections from the input ( $l = 0$ ) to the first ( $l = 1$ ) module. The total number of connections between modules is therefore

$$2N^2 r^2 \sum_{l=0}^{L-1} K_1 K_{l+1} (2/r)^{2l}.$$

For each connection, the multiply-accumulate operation requires one multiplication and one addition. The total number of connections for constant  $K_l = K$  for  $1 \leq l \leq L$  is given by

$$2K N^2 r^2 + 2K^2 N^2 r^2 \sum_{l=1}^{L-1} (2/r)^{2l}.$$

The finite geometric sum can be evaluated exactly, but we shall use the infinite sum because it converges rapidly and in any event, we are interested in the result for large Neocognitron models, so  $L$  will not be small. For large  $L$ , the sum will tend towards

$$2KN^2 r^2 + 2K^2 N^2 r^2 \frac{(2/r)^2}{1-(2/r)^2} \text{ additions, multiplications.}$$

Although we have not made any Neocognitron experiments with variable  $K_1$ , we can give a rationale for increasing  $K_1$  as  $l$  increases. With  $K_1$  held constant, the number of pathways from the  $l = 1$  module to the  $l = L$  module decreases by a factor of  $(2/r)^2$  from module to module. It may be possible to conserve the information flow by increasing  $K_{l+1}$  according to

$$K_{l+1} = (2/r)^2 K_l, \text{ for } l=1 \text{ to } L-1.$$

This is probably overly optimistic, because the information being discarded by the reduction in spatial channels is not equivalent on a one-to-one basis with pattern classification information. However, we can use this relation for the growth of  $K_1$  as an upper bound. The relation assumes the number of connections remains a constant  $K^2 N^2 r^2$  between modules  $l = 1$  to  $l = L - 1$ . So a bound on the total number of additions and multiplications required by the connections between modules is:

$$2KN^2 r^2 + 2(K^2 N^2 r^2)(L - 1) \text{ additions, multiplications.}$$



The only difference between this upper bound and the previous expression is that:

$$(L - 1) \text{ has replaced } (2/r)^2 / (1 - (2/r)^2).$$

### 3.2.2.2 Connections Within Modules

Within the  $l$ th module, there are

$$2K_1 N^2 (2/r)^{2l} r^2$$

connections between the S and C layers. The factor of 2 comes from the two types of connections, excitatory and inhibitory. There is a factor of  $K_1$  to include all of the channels, and a factor of  $N^2 (2/r)^{2l}$  which is the number of units in each C slab. Finally, the factor of  $r^2$  comes from the size of the C unit receptive fields. Each connection requires a multiply; the number of additions is precisely one less than the number of multiplications. In the limit of large  $N$ , this difference is negligible.

Summing over all modules from  $l=1$  to  $l=L$ , there are a total of

$$2N^2 r^2 \sum_{l=1}^L K_1 (2/r)^{2l} \text{ multiplications, and}$$

$$2N^2 r^2 \sum_{l=1}^L K_1 (2/r)^{2l} \text{ additions.}$$

If  $K_1$  is a constant for  $l = 1$  to  $L$ , we again take the limiting expression for the geometric series sum approximate for large  $L$  to obtain

$$2N^2 r^2 K \frac{(2/r)^2}{1-(2/r)^2} \quad \text{multiplications and,}$$

$$2N^2 r^2 K \frac{(2/r)^2}{1-(2/r)^2} \quad \text{additions.}$$

In the case of  $K_1 = K(r/2)^{2(l-1)}$  for  $1 \leq l \leq L$ , all of the terms in the sum have the same value,  $K(2/r)^2$ , so the sum is:

$$2N^2 r^2 KL(2/r)^2 = 8N^2 KL \text{ additions and multiplications.}$$

These operations occur with each activation of the feedforward processing cascade.

### 3.2.2.3 Nonlinear Combining Within Modules

Each of the  $C$  units in the  $l$ th module combines its cumulative excitatory and inhibitory inputs ( $E$  and  $I$ , respectively) in a function of the form

$$\text{LIMITER} \left[ \frac{1 + E}{1 + I} - 1 \right].$$

The LIMITER function is defined as

$$\text{LIMITER}(x) = \frac{x}{x + a} \quad \text{for } x > 0; \quad \text{LIMITER}(x) = 0 \quad \text{for } x \leq 0.$$

The inhibitory input  $I$  is computed separately for each position in the  $C$  slab, and these values of  $I$  are used by all of the  $C$  slabs in a given layer, so once the computation of  $1/(1+I)$  is made for one slab, it may be stored for the other  $K_1-1$  slabs in the  $l$ th  $C$  layer. There are  $N^2(2/r)^{2l}$  units in a  $C$  slab in the  $l$ th  $C$  layer, so this computation requires

$N^2(2/r)^{2l}$  additions and divides.

In the definition of the LIMITER function, there is an addition which may be identified as  $+a$ , and in the argument of the LIMITER function, there are additions which may be identified as  $+E$  and  $-1$ . These 3 additions are performed for all  $C$  units in the  $l$ th  $C$  layer. This involves  $K_1$   $C$  slabs in all, requiring

$3N^2K_1(2/r)^{2l}$  additions.

The multiplication of  $(1+E)$  by the prestored  $1/(1+I)$  value is also performed over all  $C$  units in the  $l$ th layer, requiring

$N^2K_1(2/r)^{2l}$  multiplications.

The LIMITER function definition involves a division,  $x/(x+a)$  which must be performed over all  $C$  units in the  $l$ th layer, requiring

$N^2K_1(2/r)^{2l}$  divisions.

Adding terms for the addition operations, the nonlinear combining operations for all modules contribute a total of

$$N^2 \sum_{l=1}^L (3K_l + 1)(2/r)^{2l} \text{ additions,}$$

$$N^2 \sum_{l=1}^L K_l (2/r)^{2l} \text{ multiplications, and}$$

$$N^2 \sum_{l=1}^L (K_l + 1)(2/r)^{2l} \text{ divisions.}$$

We are considering two types of Neocognitron architecture, those with constant  $K_l = K$  for  $l=1$  to  $L$ , and those with increasing  $K$  specified by  $K_{l+1} = K(r/2)^{2l}$ . The increasing  $K$  case provides an upper bound to the nonlinear combining operation count. Summing the above expressions for these two cases provides the following results. For constant  $K$ ,

$$(3K + 1)N^2 \frac{(2/r)^2}{1-(2/r)^2} \text{ additions,}$$

$$K N^2 \frac{(2/r)^2}{1-(2/r)^2} \text{ multiplications, and}$$

$$(K + 1) \frac{(2/r)^2}{1-(2/r)^2} \text{ divisions.}$$

In the increasing K case,

$$3KLN^2(2/r)^2 + N^2 \frac{(2/r)^2}{1-(2/r)^2} \text{ additions,}$$

$$KLN^2(2/r)^2 \text{ multiplications, and}$$

$$KLN^2(2/r)^2 + N^2 \frac{(2/r)^2}{1-(2/r)^2} \text{ divisions.}$$

#### 3.2.2.4 Nonlinear Combining Between Modules

Each of the S units in the lth module combines its excitatory and inhibitory inputs with a function of the form

$$(c) \text{RAMP} \left[ \frac{1 + E}{1 + (c/(1+c))(b)(I)} - 1 \right],$$

The RAMP function is defined as

$$\text{RAMP}(x) = x \text{ for } x > 0; \quad \text{RAMP}(x) = 0 \text{ for } x \leq 0.$$

The multiply-accumulate operations considered in Section 3.2.2.1 generate the E and I inputs to the RAMP function argument, but the operation count in Section 3.2.2.1 did not take into account the fact that the inputs to the multiply-accumulate operation for I are the squared outputs of the preceding C units in the (l-1)th module. There are  $N^2(2/r)^{2(l-1)}$  C units in each slab

in the  $(l-1)$ th module, and so there are

$$N^2 \sum_{l=1}^L K_{l-1} (2/r)^{2(l-1)} \text{ multiplications for the C unit output.}$$

In the argument to the RAMP function, the quantity  $c$  is constant within each layer, and  $b$  is constant within each slab. The product  $(c/(1+c))(b)$  may therefore be computed once per slab, and this factor and the initial factor of  $(c)$  involve 2 multiplications per S unit. Similarly, there are 3 additions, 1 division and 1 sign comparison (for the RAMP function) and 1 square root (the I inputs are RMS values) for each S unit. There are

$$N^2 (2/r)^{2(l-1)}$$

S units in each slab in the  $l$ th module, and so the nonlinear combining operations in the S layer of the  $l$ th module require

$$3N^2 \sum_{l=1}^L K_l (2/r)^{2(l-1)} \text{ additions,}$$

$$2N^2 \sum_{l=1}^L K_l (2/r)^{2(l-1)} \text{ multiplications,}$$

$$N^2 \sum_{l=1}^L K_l (2/r)^{2(l-1)} \text{ divisions,}$$

$$N^2 \sum_{l=1}^L K_l (2/r)^{2(l-1)} \text{ sign comparisons, and}$$

$$N^2 \sum_{l=1}^L K_l (2/r)^{2(l-1)} \text{ square roots.}$$

In addition we have the multiplications identified above from the squaring operations in the C layers.

We are considering two types of Neocognitron architecture, those with constant  $K_1 = K$  for  $l=1$  to  $L$ , and those with increasing  $K$  specified by  $K_{l+1} = K(r/2)^{2l}$ . The increasing  $K$  case provides an upper bound to the nonlinear combining operation count. Evaluating the above expressions for these two cases provides the following results. For constant  $K$ , the operation count for nonlinear operations between modules requires

$$3N^2K \frac{1}{1-(2/r)^2} \text{ additions,}$$

$$2N^2K \frac{1}{1-(2/r)^2} + N^2 + N^2K \frac{(2/r)^2}{1-(2/r)^2} \text{ multiplications,}$$

$$N^2K \frac{1}{1-(2/r)^2} \text{ divisions,}$$

$$N^2K \frac{1}{1-(2/r)^2} \text{ sign comparisons, and}$$

$$N^2K \frac{1}{1-(2/r)^2} \text{ square root evaluations.}$$

For the case with increasing  $K$  between modules, the operation count for nonlinear combining between modules requires

$$3N^2KL \text{ additions,}$$

$$2N^2KL + N^2 + N^2K(L-1)(2/r)^2 \text{ multiplications,}$$

$$N^2KL \text{ divisions,}$$

$N^2KL$     sign comparisons, and  
 $N^2KL$     square root evaluations.

### 3.2.2.5 Conclusions on Neocognitron Computational Load

In both the standard Neocognitron architecture, in which the number of channels  $K$  is held fixed from module to module, and the hypothetical architecture in which the number of channels increases as  $K_l = K(r/2)^{2(l-1)}$  for  $1 \leq l \leq L$ , the dominant computation is the multiply-accumulate operations which support the connections between modules. For constant  $K$ , the operation count summed over  $l$  for these connections is:

$$2N^2r^2K + 8N^2K^2 \frac{1}{(1-(2/r)^2)} = 2N^2r^2K + 8N^2K^2 \begin{array}{l} \text{additions,} \\ \text{multiplications.} \end{array}$$

For typical values of  $r = 5$  and  $K = 100$ , these terms can be combined to give the approximate growth in computation with the parameters  $N$  and  $K$  as either

$$8.5N^2K^2 \text{ or } 400N^2K \text{ additions, multiplications.}$$

For the increasing  $K$  case, there are  $2N^2K^2r^2L$  additions and multiplications, with  $L$  growing as  $\log_2(N)/\log_2(r/2)$ . (See Section 3.2.1.) Using the same values of  $r = 5$  and  $K = 100$  as before, this result may be written for comparison with the



constant K case as approximately

$50N^2K^2\log_2(N)$  or  $5000N^2K\log(N)$  additions, multiplications.

This comparison shows that the hypothetical increasing K architecture has a computational load which is greater than that for the standard Neocognitron architecture by a factor of roughly  $6\log_2(N)$  to  $12\log_2(N)$ .

Since the operation counts for both architectures grows in a reasonable rate with the number of pixels in the input window,  $N^2$ , the Neocognitron is attractive for practical values of N, say 128 or 256. Scaling up the  $N = 14$  model which we demonstrated on an IBM PC AT to  $N = 128$  will increase the computations by roughly 100x. A response time on the order of a second for the  $N = 128$  Neocognitron will require 100,000x the processing speed of the PC AT. This is feasible with parallel processing.

#### 4.0 AC-coupled Retina with Cooperative Receptors

Today's supercomputers are in many respects greatly inferior to the brain. Language learning without instruction, fluid, content-addressable memory and the natural ability of humans to perform face recognition are examples of the superiority of the brain. The capabilities of the brain are all the more remarkable when its size, weight and power requirements are compared to those of computers.

These attributes of the brain compel us to try to understand its form and function. A natural place to start is the most accessible part of the brain, the retina.

The human visual system is adaptive in many diverse ways. For example, we perceive colors fairly well even though the spectrum of the illuminant changes sharply when going indoors or outdoors. While reading, we adapt to different fonts and type sizes. We "see" fairly well under both moonlit and sunlit conditions, and we frequently must drive cars with sunlight or headlights shining in our eyes.

In the "computer-vision" style of information processing, the front-end of the system, a camera and analog-to-digital convertor, is selected to capture the scene intensity and spectrum faithfully for subsequent processing. Not so in the biological style of information processing. Even at the

photoreceptor level, our visual system is responding to the relative image brightness, rather than the absolute brightness. In solid-state focal plane array cameras, "fixed pattern noise" caused by inhomogeneities in the detector material or fabrication must be corrected by special "field flattening" hardware or algorithms. Not so in the human visual system. Yes, the retina has inhomogeneities-- like the blood vessels which obstruct the retina, but we don't see fixed pattern noise because the retina only responds to changes. Eye tremor provides the necessary modulation to refresh the scene on the retina, while the AC-coupling of the retina filters out fixed pattern noise.

The retina is part of the brain, and it is likely that the style of information processing known to occur in the retina is used elsewhere in the brain for diverse functions. For example, the natural ability of the retina to eliminate fixed pattern noise can be recognized more generally as a form of delta modulation to achieve bandwidth compression and maximize information flow.

The retina is therefore significant both as a new, biological paradigm for imaging sensors, and for the opportunity it provides to learn about the brain. These two reasons provided the motivation to study a retinal model developed by Michael H. Brill for his Ph. D. dissertation at MIT under Professor J. Y. Lettvin. We improved the computational approach used in the model (called IRIS) so that it became possible for the first

time to scale up the retina to interesting array sizes (up to 256x256). We did this first in FORTRAN on an IBM PC AT, and later on a SUN Workstation, and a BBN Butterfly machine. The BBN Butterfly is a parallel machine. Using 16 processors on the Butterfly, the model ran 25.14x faster than on the SUN. This work is reported separately in an appendix, PARALLEL IMPLEMENTATION OF IRIS. Although this is a significant speedup, the Butterfly is a coarse-grained parallel machine, and the IRIS retinal model involves local processes, suggesting that it is most naturally implemented as a special purpose analog or digital chip, as Carver Mead has done with his retinal model. ("A Silicon Model of Early Visual Processing," by Carver Mead and M. A. Mahowald, Neural Networks, Vol. 1, pp91-97, 1988.)

The IRIS retinal model has the capability to adjust spatiotemporal resolution to the prevailing light level, thereby combating photon noise. This property is known as adaptive resolution. It also has an automatic gain control (AGC) capability to adapt the set-point of dynamic range to the local average light intensity. This property is known as adaptive contrast resolution.

IRIS receptors have three-state activation kinetics with short-lived intermediate photopigment states. The model is "AC-coupled" in the sense that only temporally varying light/dark gradients maintain a sensory excitation. The response of the receptor cells is coded as voltage, and the

receptors are resistively coupled. This arrangement provides an automatic mixture of time/space integration: a) for high photon flux, each receptor cell responds rapidly, and the neighboring cells are decoupled, providing the highest spatiotemporal resolution; b) for low photon flux, cell temporal response is slow, and neighboring cells are coupled to give broader spatiotemporal integration, enhancing light sensitivity at the expense of resolution in space and time.

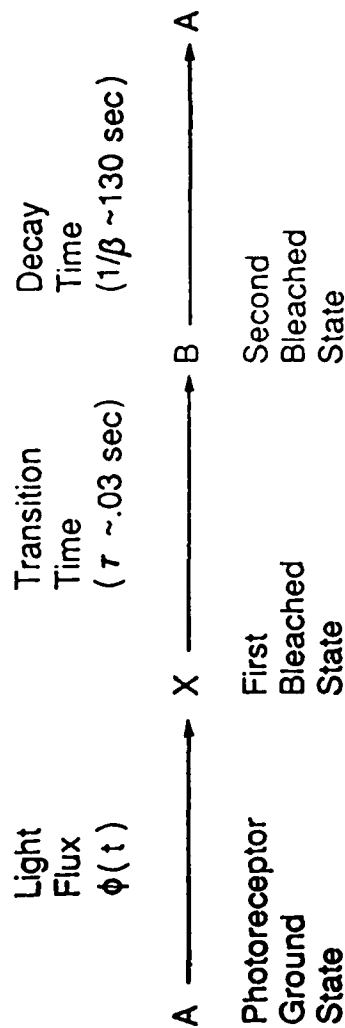
In the model retina, transduction of light into receptor response is the result of ionic photoconductors embedded in the receptor cell membrane. Depending on its instantaneous state, each photopigment molecule in the membrane can open a conducting channel to either of two monovalent cationic species. When a photon is incident, the receptor quickly opens a channel to the first of these species, then slowly closes this channel and opens a channel to the other species, and finally returns to a nonconductive state (by the addition of metabolic energy). The ionic species are driven by membrane voltages in opposite directions, so the receptor acts in a "push-pull" way. As in human vision, the voltage response is zero to a sustained (steady-state) light; the response versus log-intensity function shifts to the time-averaged intensity, thereby giving the adaptive contrast sensitivity desired for a visual system with limited dynamic range. Also, the receptor's response is governed by photopigment kinetics whose rate increases with light level. Hence, the model retina has adaptive temporal

resolution-- as desired to defeat photon noise. The photopigment kinetics are described by a set of coupled first-order differential equations provided in Figure 12.

The adaptive time behavior of the model receptor has another helpful property borne out in our computer simulations. A pulse of light causes a response that decays slowly to steady state; however, a pulse of darkness produces only a short-duration change of response. This is fortunate for an eye that has to blink. See Figure 13 for simulations of the model receptor response to changes in light level.

The adaptive temporal resolution of the model retina has a simple counterpart in the spatial domain. The model retina is tiled with a lattice of photoreceptors, wired together with a passively conducting grid of constant conductivity. The visual signal for each receptor --a transmembrane voltage-- is now modified by lateral interaction. See Figure 14. When the retinal illumination is (and has been) very low, most of the current passes between receptors when light hits one of them, and the receptors are functionally coupled. However, when the eye is light-adapted, the receptors tend to keep to themselves; very little current flows between them. This behavior shows up as an "iris" of lateral influence, contracting in the presence of light and dilating when light is removed. It provides a simple mechanism whereby visual acuity can be traded off against light sensitivity as the prevailing light gets dimmer.

# Photopigment Population Transition Model



- First-order kinetic model with light flux  $\phi$  as a coefficient

- Differential Equations

$$\dot{A}(t) = -\phi(t) A(t) + \beta B(t)$$

$$\dot{X}(t) = \phi(t) A(t) - \phi(t - \tau) A(t - \tau)$$

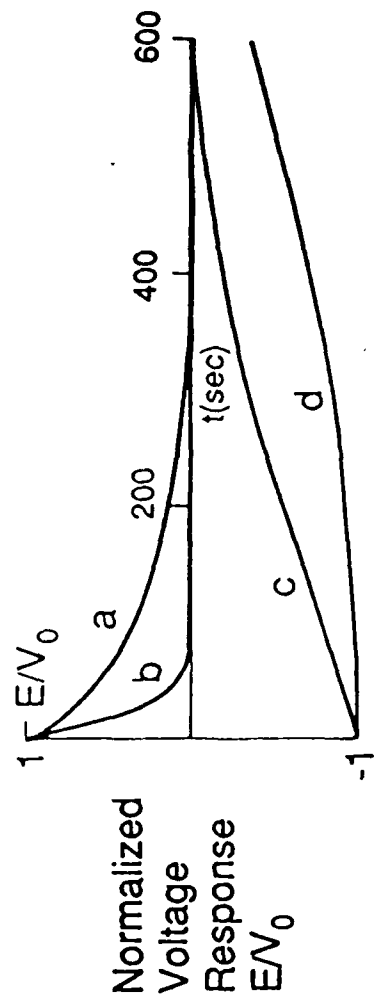
$$\dot{B}(t) = \phi(t - \tau) A(t - \tau) - \beta B(t)$$

$$A + B + X = C = \text{const.} \approx 10^8$$

- A, B, X are populations of photopigment molecules in a cone receptor

Figure 12

# Receptor Responses to Light Steps



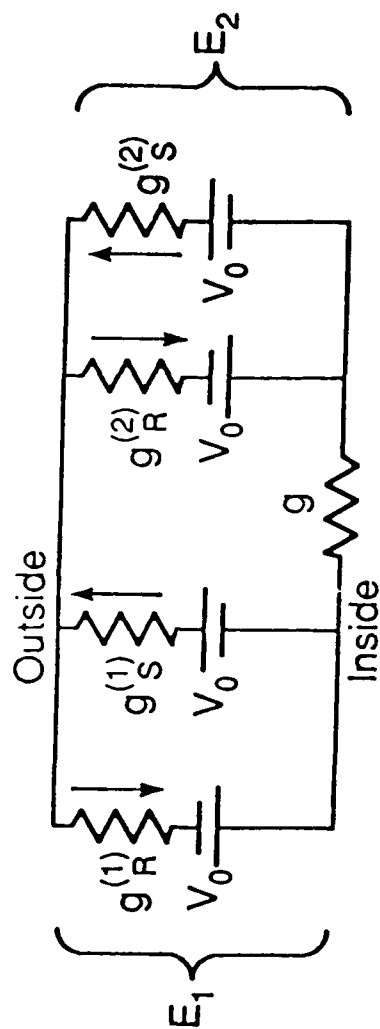
	Initial Flux	Final Flux	
a	$10^{-5}$	$10^{-3}$	Positive Step Input
b	$10^{-3}$	$10^{-1}$	
c	$10^{-1}$	$10^{-3}$	Negative Step Input
d	$10^{-3}$	$10^{-5}$	

Flux units are photons/molecule/sec (ppms)

Figure 13.



## Spatial Coupling of Two Receptors



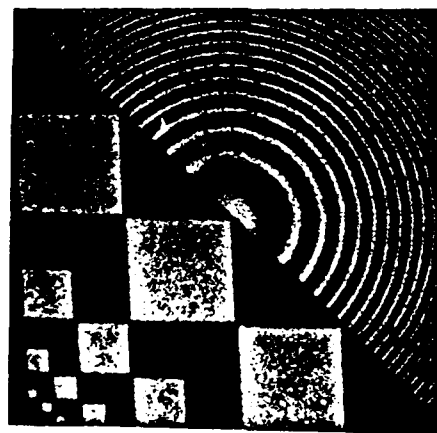
- Lateral conductance  $g = \text{constant}$
- $g_R, g_S$  increase with increasing  $\phi$
- If  $\phi$  large  $\Rightarrow g \ll g_R^{(K)}, g_S^{(K)}$   
 $\Rightarrow$  uncoupling in bright light

Figure 14.

Results of an IRIS simulation provided in Figure 15 demonstrate this intensity-dependent spatial resolution. The test pattern is split into two portions along the diagonal. Above the diagonal, a pattern of black and white squares recedes into the upper left corner. Below the diagonal, there is a cosinusoidal pattern with linear FM modulation in the radial direction (a Fresnel pattern). The limiting resolution is determined by the discrete photoreceptor sampling at high light levels. This is demonstrated in Figure 15a (where the peak light level is 0.12 of that level which would bleach ten percent of the photopigment into a state requiring a metabolic input before returning to the ground state). At one tenth this light level, (see Figure 15b) a small decrease in resolution is evident because of spatial averaging among nearby photoreceptors. A further ten-fold decrease in the light level results in the obvious reduction in spatial resolution shown in Figure 15c. Such intensity dependent resolution is also found in human vision.

In common with human vision, the model retina displays Weber's law (proportionality of increment threshold to a pre-existing background intensity over a much greater intensity range than the instantaneous dynamic range of the receptor).

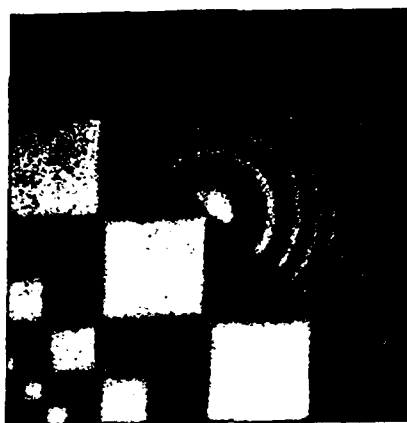
IRIS was designed to give repeatable response in dim light to visual scenes that are nominally the same except for photon noise. The repeatability was brought about at the expense of spatiotemporal resolution. Repeatability of percepts from the



**a**



**b**



**c**

Figure 15. IRIS simulation results. (a) The highest resolution is obtained for high light levels. (b) At one-tenth the light level in (a), some loss of resolution is perceptible. (c) A further ten-fold decrease in intensity results in noticeable loss in resolution.

same reflecting objects is a necessary but not sufficient condition for lightness constancy and color constancy --the illuminant-invariant assessment of reflectance information.

Further detail on the IRIS model may be found in the paper "Retinal model with adaptive contrast sensitivity and resolution," by Michael H. Brill, Doreen W. Bergeron, and William W. Stoner in the December 1, 1987 issue of Applied Optics, a special issue devoted to neural networks edited by Gail Carpenter and Stephen Grossberg.

## 5.0 Bibliography

1. Michael H. Brill, Doreen W. Bergeron, and William W. Stoner, "Retinal model with adaptive contrast sensitivity and resolution," *Applied Optics*, 26 #23, pp 4993-4998, 1 Dec 87.
2. K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biol. Cybernetics*, Vol. 36 pp 193-202, 1980.
3. Paul Horowitz and Winfield Hill, "The Art of Electronics," p117 logamps, p41 and p120 active clamps, Cambridge University Press, Cambridge England, 1980.
4. Hisatoyo Kato and J. W. Goodman, "Nonlinear filtering in coherent optical systems through halftone screen processes," *Applied Optics*, 14 #8, pp1813-1824, Aug 75.
5. D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proc. Royal Soc. Lond.* B204, pp301-328, 1979.
6. Carver Mead and M. A. Mahowald, "A Silicon Model of Early Visual Processing," *Neural Networks*, 1, pp91-97, 1988.
7. William Stoner and Terry M. Schilke, "Pattern Recognition with a Neural Net," *SPIE Vol 698, Real Time Signal Processing IX* pp 170-181, 1986.

6.0 APPENDIX

PARALLEL IMPLEMENTATION OF IRIS

Doreen Bergeron

(SAIC)

Tucson, Arizona

July 17, 1987

This document describes the implementation of the fortran program, COLORP, on the Butterfly multiprocessor. The COLORP program is a computer simulated trichromatic retina ( called IRIS ) as described in Retinal Model with Adaptive Contrast Sensitivity and Resolution by Michael H. Brill, Doreen W. Bergeron, and William W. Stoner. The purpose of this experiment was to show the reduction in resolution of the output image as corresponding input image light levels were reduced, and to increase the runtime speed of the program by implementing on a parallel processor.

An unoptimized version of the code was tested on the SUN uniprocessor which did produce output images that demonstrate the reduction in the spatiotemporal resolution as the light levels were reduced. Table 1 summarizes the results of the 7 test runs performed on the SUN. Column 1 is the run number. Columns 2 and 3 list the initial phi and intensity values used. The initial phi value is used to calculate the initial values of the three-phase receptors and is one-half that of the maximum intensity value. Then listed is the chip size, min and max input values, and min and max of the resulting output arrays. Runs 2, 3, and 4, which showed the greatest difference in the resolution of the output arrays, were rerun on a 256 x 256 image and photographed. The results are shown in figure 1.

Figure 2 is a diagram of the calling sequence of the uniprocessor version of the program. Each routine is briefly described below:

COLORP - Driver which controls flow of the program and produces jitter effect.

PICL1 - Provides an  $n \times n$  picture as input to the retinal array. The pattern lying above the diagonal is a series of squares decreasing in size. The pattern lying below the diagonal is a cosinusoidal fresnel zone plate.

KNUTH1 - Random number generator used to determine jitter effect.

BLEACH - Finds change in photo-chemical species of three-phase receptors as functions of quantum cath phi.

SYNADI - Performs syncytial lateral interaction by using the alternating direction implicit (ADI) solution of partial differential equations.( from Numerical Recipies)

TRIDAG - Solves tridiagonal system of equations.(from Numerical Recipies)

IMGOUT - Writes an image out to a RAMfile.

RAMGLUE - Fortran-C Interface to BBN ramfile system ( Mike Ingram SAIC Tucson)

RAMFILE SYSTEM - File handler written to maintain ramfiles on the Butterfly system. (BBN) The RAMfile system was simulated on the SUN.

The SUN version of the program was modified in the following ways in order to optimize the code and minimize its size for implementation on the Butterfly.

- Only one call to synadi was made rather than twelve.
- Portion of code which produces jitter indices was replaced by a do loop.
- Bleach routine was replaced by a worker routine as were two sections of synadi which set up input vectors to the tridag routine and call that routine.
- All arrays referenced in the worker routines were scattered throughout butterfly memory in order to reduce contention.
- All frequently referenced variables appearing in worker routines were shared among processors ( local processor private copies were made .)
- Output data word size was reduced to integer\*2 because values fall between 0 and 255.
- Jitter indices are passed to the bleach routine, eliminating the need for an additional array.
- Third dimension of A array in bleach routine was replaced with a temporary variable.
- All arrays were reduced to two dimensional in order to be able to allocate them.

#### Parallel Implementation :

Three portions of the code were identified as targets for parallel implementation. The first is the bleach routine, which finds the changes in the photochemical species of three-phase receptors by computing coupled differential equations for each pixel. A generator calls the bleach worker routine once for each element of the image array thus a GENONA call is used.

Two worker routines were extracted from the synadi routine.

1.) Body1 which computes input vectors for the tridag routine and computes a solution for PSI, one line at a time.



2.) Body2 which also computes vectors for the tridag routine and computes a solution for V and PSI one line at a time.

A generator calls these routines once for each row of the image thus a GENONI call is used.

In order to measure the timing of the code on various node configuration, it had to be reorganized. The new calling scheme is shown in figure 3 and the new subroutines briefly described below.

TIMECOLORP - Driver for the timed version of the program.

INITPROBONCE - Contains all memory allocation and matrix scattering calls. Makes the call to pic11 to produce the input array.

INITPERRUN - Initializes arrays to pre-bleached state.

EXECUTE - Portion which does the actual computation. It performs all generator calls. This is the only portion of the program which is timed.

PRESULTS - prints results of time tests.

The program was run on every possible number of nodes ( 1 - 26 ). The timing routine reports the time it took the code to execute, the number of processors it used, the effective number of processors, and the efficiency. The effective number of processors is equal to the time it takes one processor divided by the time it takes n processors ( n is the number of processors being used.) The efficiency equals the effective number of processors divided by n processors.

A summary of the testing results is shown in table 2. Tabulated is the number of processors, time in seconds that it took the algorithm to run, effective number of processors, and the efficiency. Figure 4 is a plot of the time it took the program to run versus the number of processors it ran on. Note that the time it took to run on one processor was 4137.98 seconds, and that was decreased by a factor of 7 when the number of processors was increased to 2. The significant decrease can partially be attributed to the fact that even though only one processor is being used to run the program, the data is scattered across all available nodes ( 26 of them.) The one processor must make many switch calls to access the data which is too numerous to fit on one node thus producing an I/O bottleneck. There may also be some unidentified overhead computation involved in the initial run of the routine, perhaps some memory management initialization which is not locatable at this high level of testing. Figure 5 is a rescaled version of figure 4 with the first data point eliminated. The fluctuation in runtime is more apparent in this plot as is the tendency of the runtime to decrease as the number of processors approaches 16. The runtime then increases as more processors are added above 16. The index generator calls to the

worker routines Body1 and Body2 were made 256 times and the array generator calls were made on a 256 x 256 array, so it is not surprising that the statistics show the program running faster and most efficiently on a configuration of 16 nodes.

Figure 6 is a plot of the computed effective processors and a linear plot. It is interesting to note that the 2 curves intersect at the data point for 17 processors. The effective processor curve reaches a maximum value of 17 at the 16 processor point. The data also shows that 11 processors would be an efficient number to work with for this algorithm.

Figure 7 is a plot of processor efficiency vs number of processors. The efficiency steadily decreases after the efficiency peak at the two processor point.

Timing was also performed on a comparable version of the SUN code. There were slight differences between the data structures used in the two versions, but nothing which would amount to any major processing time differences. Processing of a 256 x 256 image on the SUN took 1:37:02 or 5822 seconds. When compared with the single processor run on the Butterfly, the SUN took  $(5822/4138) = 1.4$  times longer to run. In the two processor case the butterfly was  $(5822.0/581.3) = 10.0$  times faster than the SUN. The greatest speedup factor occurred in the 16 processor case which ran  $(5822.0/231.58) = 25.14$  times faster. There still remains the discrepancy between the numbers of the first two timing runs performed on the butterfly. More rigorous timing procedures may be able to identify a source of overhead computation involved in the first run of the timing series on the Butterfly.

Table 1

## SUMMARY OF 7 SUN RUNS

RUN	PH11	INTENSITY	SIZE	INPUT-RANGE	V-RANGE	E-RANGE
1	1.5e-04	1.0e-03	32	1.01e-03 1.00e-05	0.712274 -0.753071	0.728196 -0.881850
output very bright						
2	5.0e-04	1.0e-03	32	1.01e-03 1.00e-05	0.337748 -0.960784	0.337009 -0.960881
slightly blurred image						
3	5.0e-05	1.0e-04	32	1.01e-04 1.00e-06	0.314754 -0.654061	0.337013 -0.960881
blurred image						
4	5.0e-06	1.0e-05	32	1.01e-05 1.00e-07	0.160709 -0.148879	0.337014 -0.960880
very blurred image						
5	5.0e-07	1.0e-06	32	1.00e-06 1.00e-07	no output	
maxits exceeded in synadi ( maxits = 1000 )						
6	5.0e-03	1.0e-02	32	1.01e-02 1.00e-04	0.332361 -0.959805	0.336964 -0.960886
fairly sharp image						
7	5.0e-02	1.0e-01	32	1.01e-01 1.00e-03	0.335810 -0.959701	0.336513 -0.960939
very similar to output of run 6						

<u>processors</u>	<u>time</u> <u>(secs)</u>	<u>effective</u> <u>processors</u>	<u>efficiency</u>
1	4137.98	1.00	1.00
2	581.3	7.12	3.56
3	428.88	9.65	3.22
4	354.8	11.66	2.92
5	308.99	13.39	2.68
6	279.48	14.81	2.47
7	261.59	15.82	2.26
8	258.2	16.03	2.00
9	243.02	17.03	1.89
10	244.67	16.91	1.69
11	233.12	17.75	1.61
12	241.43	17.14	1.43
13	242.67	17.05	1.31
14	237.73	17.41	1.24
15	243.3	17.01	1.13
16	231.58	17.87	1.12
17	244.37	16.93	1.00
18	255.03	16.23	0.90
19	253.58	16.32	0.86
20	244.61	16.92	0.85
21	271.61	15.24	0.73
22	257.11	16.09	0.73
23	273.68	15.12	0.66
24	270.42	15.30	0.64
25	261.34	15.83	0.63
26	290.43	14.25	0.55

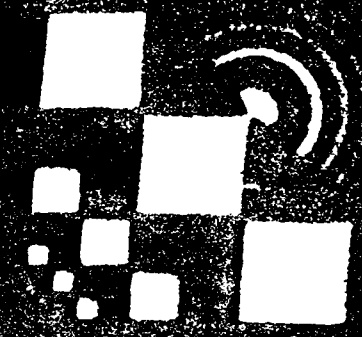
Table 2 Time Test Results



lag2. 2. sun



lag2. 3. sun



lag2. 4. sun

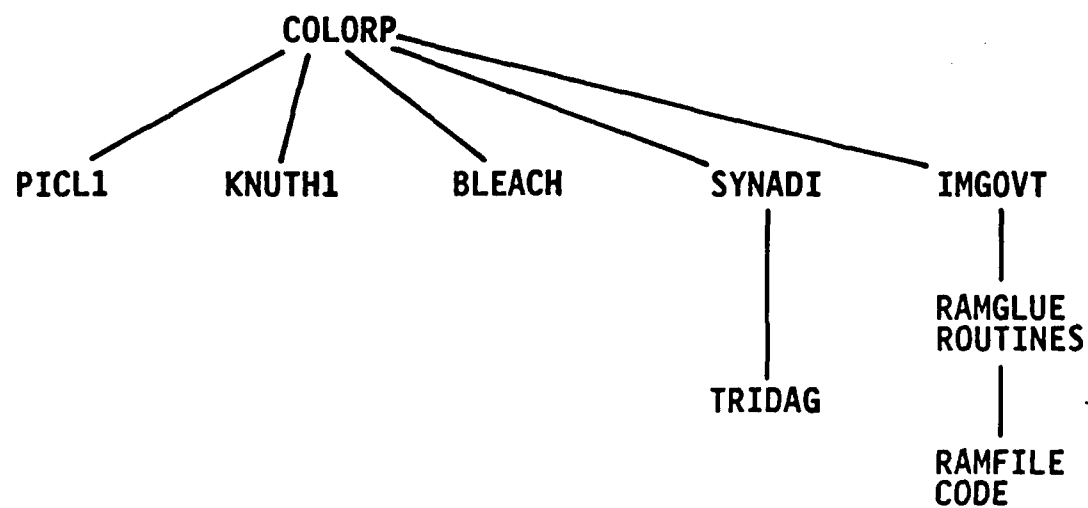


Figure 2

G-244-87

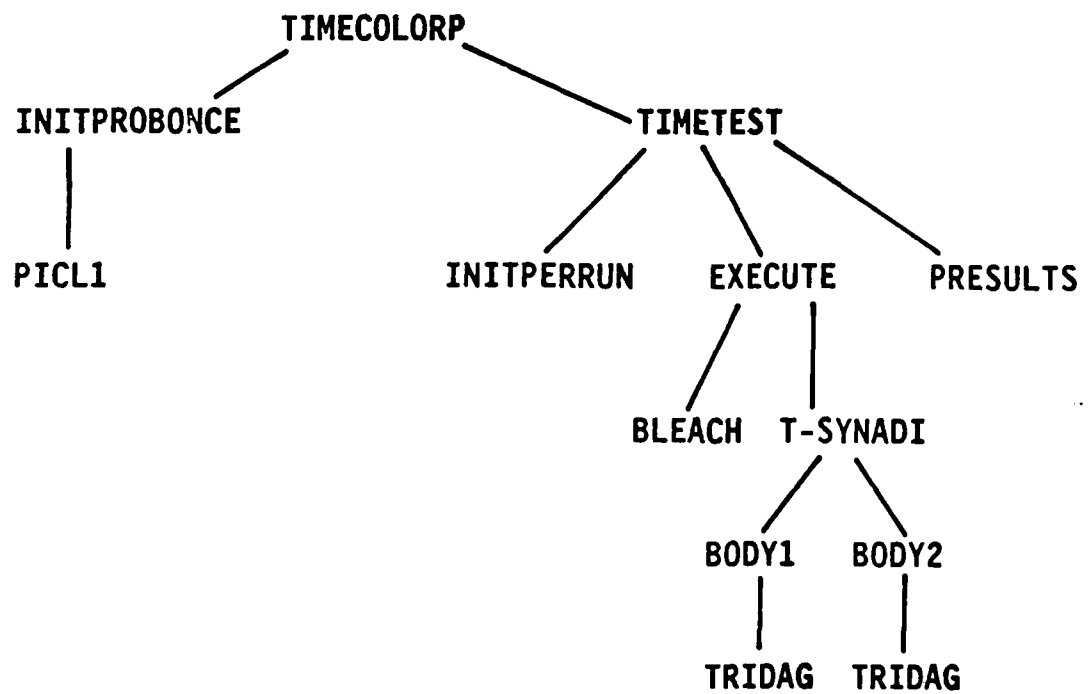


Figure 3

G-243-87

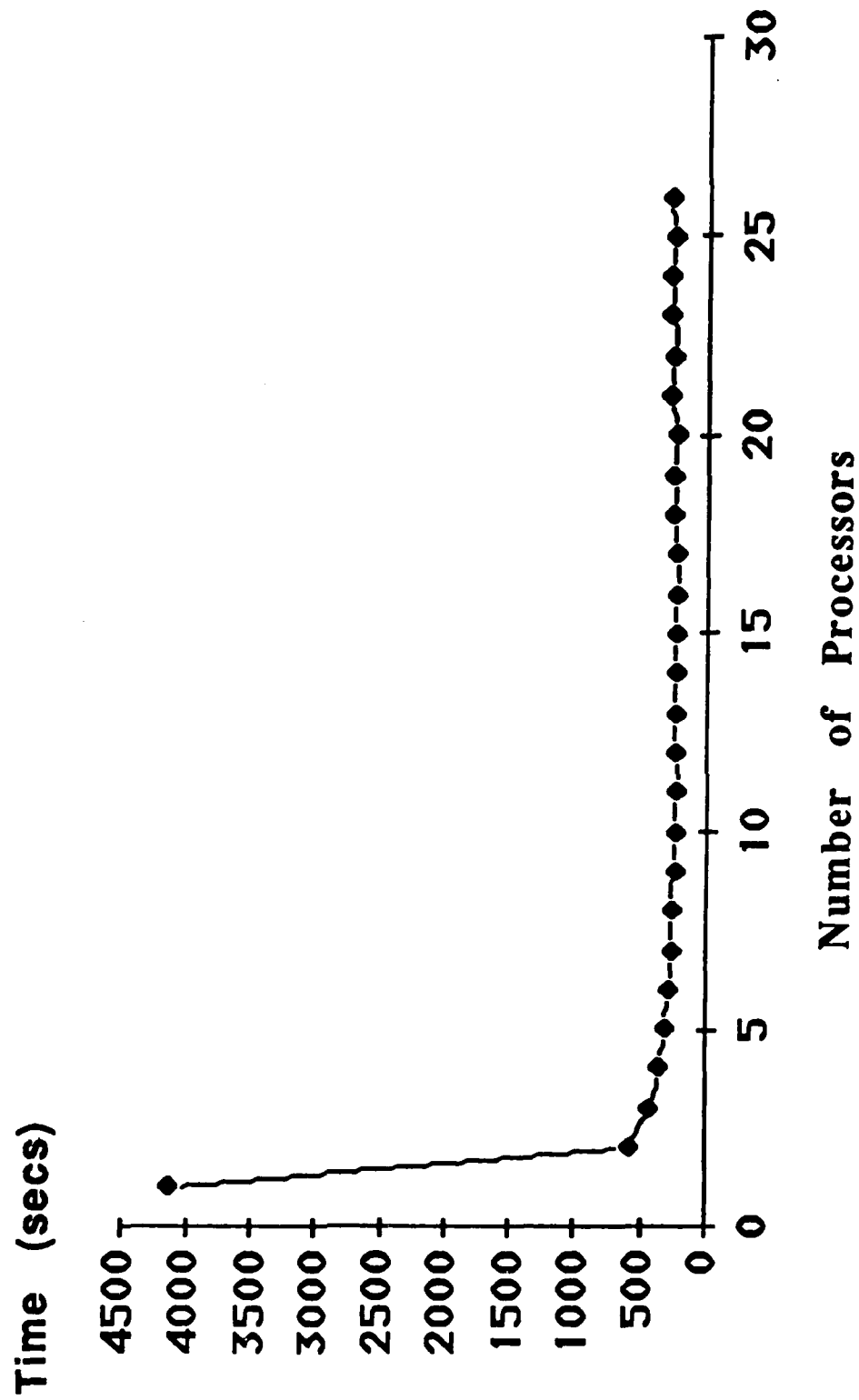


Figure 4 Runtime vs Number of Processors



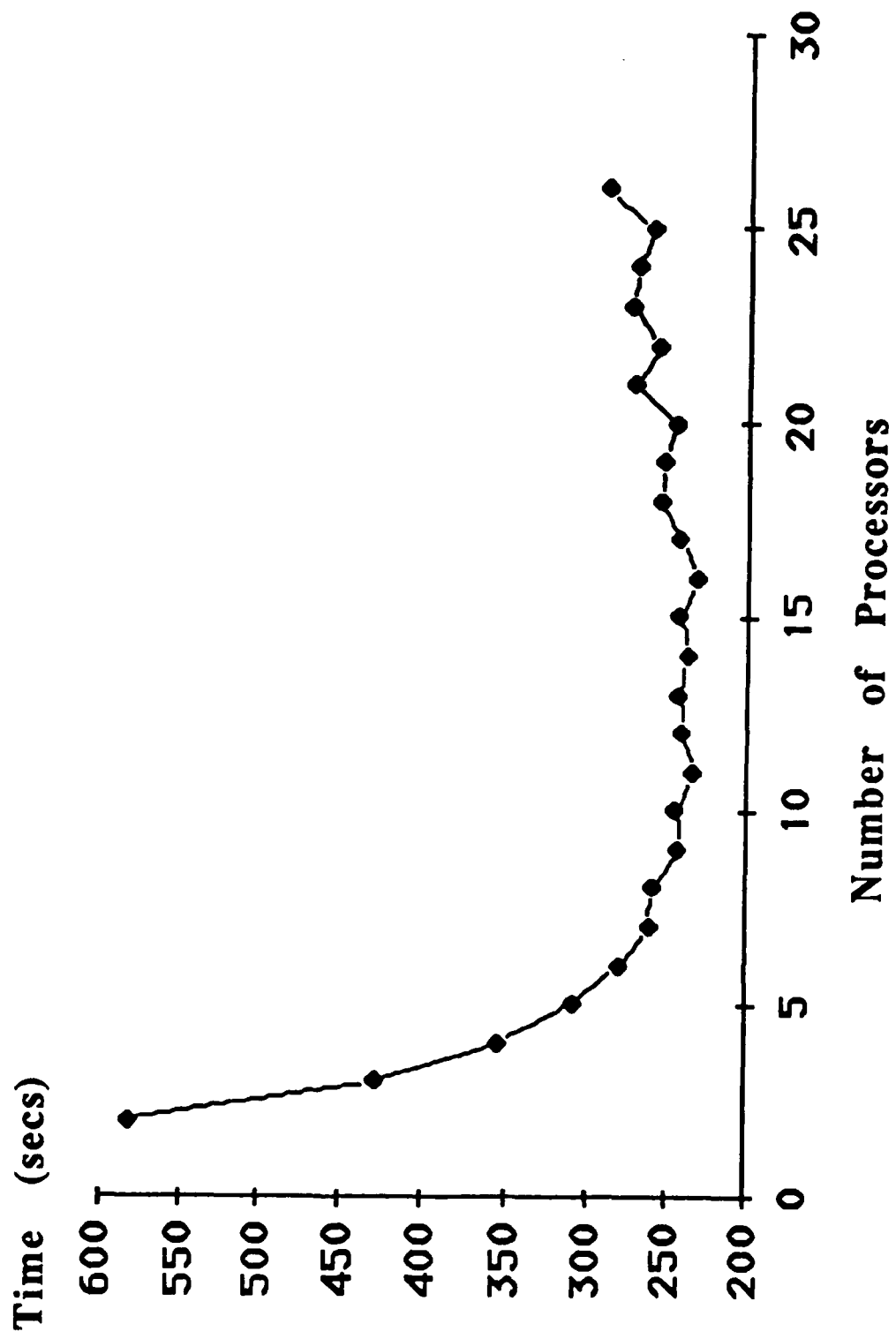


Figure 5 Runtime vs Number of Processors

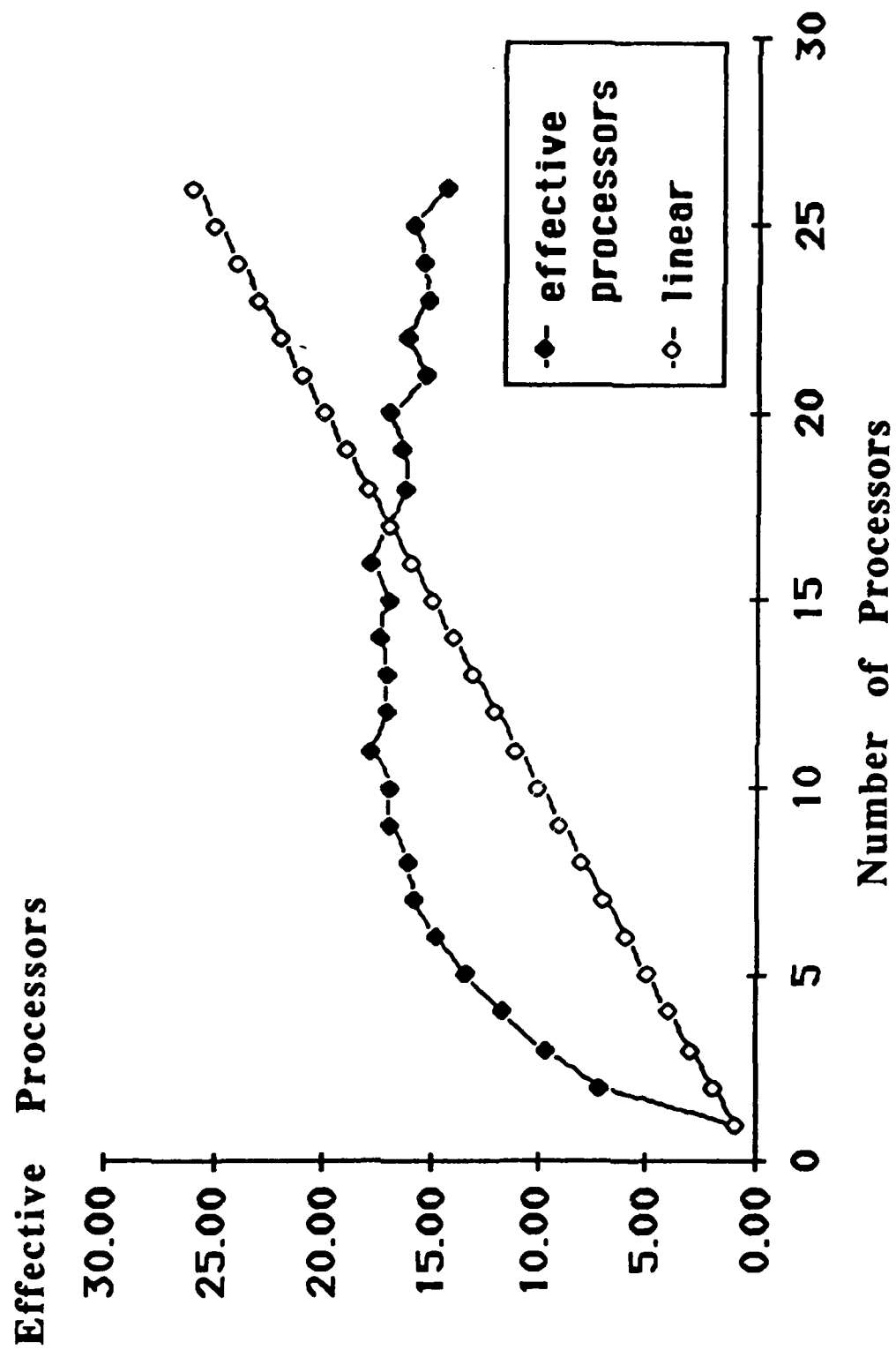


Figure 6 Effective Number of Processors

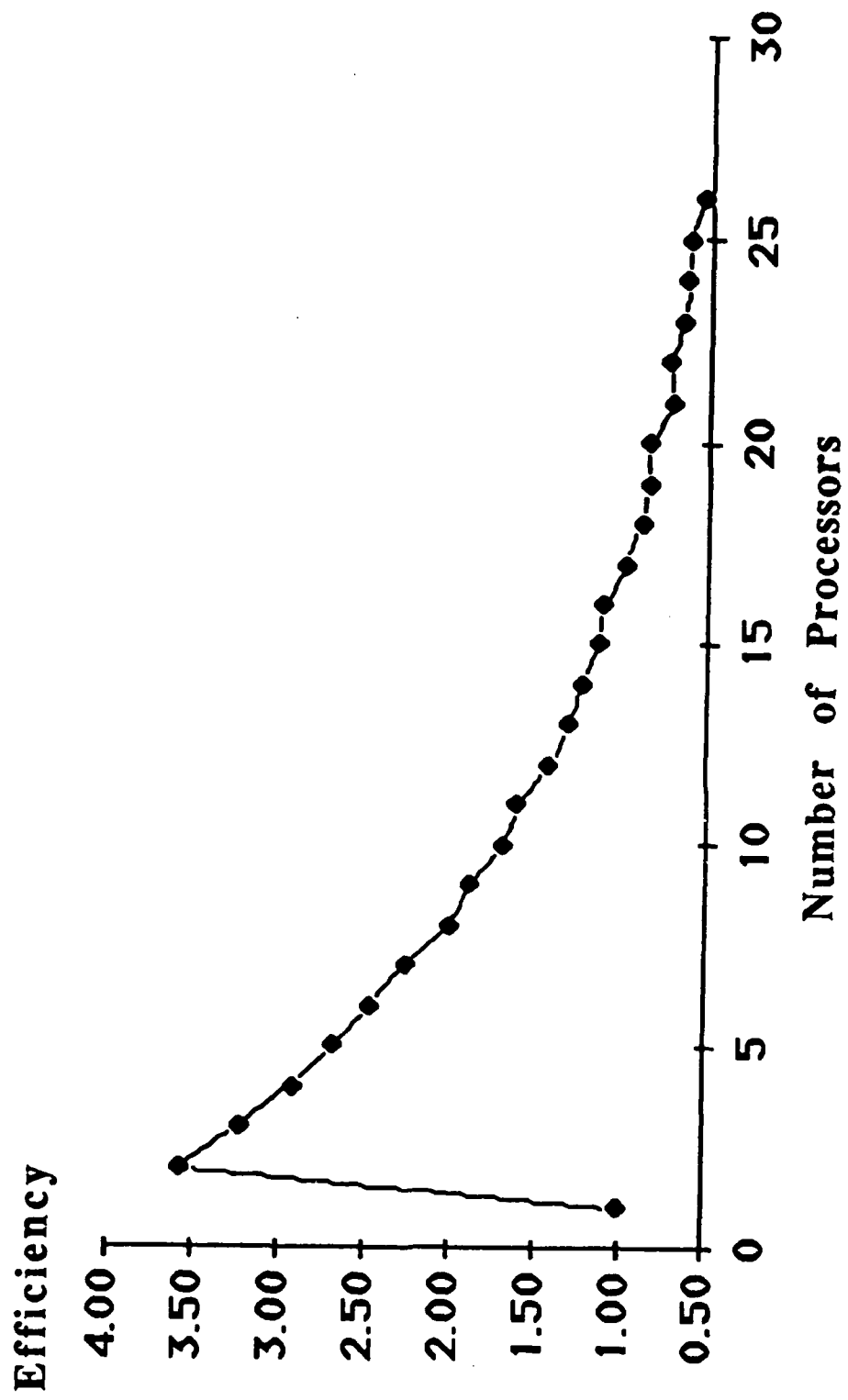


Figure 7 Processor Efficiency vs Number of Processors